IBM Application Program

H20-0241-3

1130 Commercial Subroutine Package

(1130-SE-25X), Version 3

Program Reference Manual

The IBM 1130 Commercial Subroutine Package is for IBM 1130 users with a knowledge of FORTRAN. The package is not intended to make FORTRAN a complete commercial language, but to supply commercial capability to users of IBM 1130 FORTRAN.

This manual is a combined user's, operator's, and system manual.

Fourth Edition

This edition, H20-0241-3, is a major revision obsoleting H20-0241-2.

A form is provided at the back of this publication for reader's comments.
If the form has been removed, comments may be addressed to IBM Corporation,
Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601

CONTENTS

INTRODUCTION

The 1130 Commercial Subroutine Package has been written to facilitate the use of FORTRAN in basic commercial programming. Included in the package are the following items:

- The GET routine, which allows the programmer to decode input records after they have been read. This eliminates the common FORTRAN-associated problem that occurs when input cards enter the system in an unknown sequence. Input records that vary in this way may be read with the A1 format and converted to real numbers (using GET) after the program has determined which type record was just read.

- An editing routine, EDIT, for the preparation of output in special formats. With EDIT it is possible to insert commas, supply leading blanks, float dollar signs, display a CR symbol after negative numbers, etc. EDIT is especially useful in the preparation of invoices, checks, and other commercial documents.

- Code conversion routines for data manipulation and more efficient data packing:

GET	-	A1 format to Real
PUT	-	Real to A1 format
PACK	-	A1 to A2 format
UNPAC	-	A2 to A1 format
A1A3	-	A1 to A3 format
A3A1	-	A3 to A1 format
DPACK	-	D1 to D4 format
DUNPK	-	D4 to D1 format
A1DEC	-	A1 to decimal format
DECA1	-	Decimal to A1 format

- A variable-length decimal arithmetic package. In this system, all arithmetic is done with integer or decimal numbers, with field lengths chosen by the user. This subset of the Commercial Subroutine Package includes routines for variable-length decimal add (ADD), subtract (SUB), multiply (MPY), divide (DIV), compare (ICOMP), and sign test (NSIGN).

  Use of this system eliminates two of the arithmetic problems associated with FORTRAN: the accuracy problem (the inexact representation of fractions) and the magnitude problem (extended precision values limited to nine digits, etc.).

- Subroutines for improved speed and control of I/O devices. By taking advantage of the 1130's cycle-stealing capability, the overlapped I/O routines can substantially speed the throughput rates of many jobs. Subroutines are supplied for the

  IBM 1442 Card Read Punch
  IBM 1442-5 Card Punch
  IBM 2501 Card Reader
  IBM 1132 Printer
  IBM 1403 Printer
  Console Keyboard
  Console Typewriter

In addition to input/output, subroutines are supplied for control of the 1132 and 1403 carriage and the 1442 stacker select mechanism.

- Several utility routines for common tasks:

NCOMP	for comparing two variable-length alphameric (A1) fields
MOVE	for moving data from one area to another
FILL	to fill an area with a specified value
WHOLE	to truncate the fractional portion of a real number
NZONE	for testing and modifying zone punches

USE OF THE COMMERCIAL SUBROUTINE PACKAGE

CSP is modular in design -- the user may use whichever routines he needs and ignore the others.

The routines may be assembled on any 4K card 1130 system, but an 8K system will probably be required for any extensive usage. The desired subroutines may be inserted in the FORTRAN execute deck (card systems) or stored in the Subroutine Library on the disk cartridge. In addition, some of the CSP routines use certain parts of the IBM 1130 Subroutine Library. (See "Core Allocation" in the Appendix.)

All of the routines are written in the 1130 Assembler Language.

The control statement

```
*ONE WORD INTEGERS
```

must be used in programs that call any of the Commercial subroutines.

The control statement

```
*EXTENDED PRECISION
```

must be used in any program that calls the GET or PUT subprograms. The other CSP routines are independent of the real number precision.

In general, CSP will operate under either Version 1 or Version 2 of the 1130 Disk Monitor System. The exceptions are P1403, S1403, P1442, and R2501, which use subroutines supplied only with Version 2 (see the detailed descriptions for more particulars).

The use of the overlapped I/O portion of CSP is an "either/or" proposition. For nondisk I/O, the programmer must choose either the CSP overlapped routines or the standard FORTRAN routines. The two systems cannot be intermixed within the same program. Note the emphasis on nondisk. This exclusion does not apply to disk I/O, which may be used regardless which of the two systems is selected.

Use of the overlapped I/O routines also excludes the employment of the TRACE feature of FORTRAN, since it used portions of the FORTRAN package for output.

MACHINE REQUIREMENTS

For execution, an 8K 1130 system, with any card reader, is necessary. In addition, the following I/O devices are supported:

1442 Card Read Punch, Model 6 or 7
1442 Card Punch, Model 5
2501 Card Reader, Model A1 or A2
1403 Printer, Model 6 or 7
1132 Printer
Console Keyboard
Console Typewriter

Other I/O devices may be utilized through standard FORTRAN.

For assembly, any 1130 card system is sufficient. The subroutines may be card- or disk-resident.

SPECIAL CONSIDERATIONS — ARITHMETIC

Real arithmetic. When using CSP, remember that the standard FORTRAN limitations apply to all real numbers.

Extended precision numbers should not exceed ±1,000,000,000. (or 9 digits).

Fractions must be avoided if exact results are desired. All critical arithmetic should be done with whole numbers. For example, the extension

40.75 hours x $2.225 per hour

should be carried out as

4075. hundredths of hours x 2225. mills per hour

If this is not done, precision errors may appear in the results.

Decimal arithmetic. If the nine-digit or fractional limitations of FORTRAN prove burdensome, the Decimal Arithmetic package may be used. In this system, all arithmetic is done with whole numbers (no fractions), and the number of digits in each variable is chosen by the user.

A number in decimal format may be as long as desired; there is no practical limit to field length.

SPECIAL CONSIDERATIONS — INPUT/OUTPUT

FORTRAN FORMAT I/O

In general, CSP works with arrays in A1 format -- one alphameric character per word. For those routines that operate on other formats, conversion routines are supplied to ease the translation between A1 and the other format.

In this area, however, one complication may occur: the use of zone punches. In many commercial applications, it is customary to X-punch the units position of a credit or negative field. Because the 11-0 Hollerith combination is not recognized by the conversion routines used with FORTRAN READs, it is necessary, when keypunching, to omit the 0-punch when an 11-punch is present in the same column. This is not a problem with 1130-produced cards that later serve as input to subsequent runs. No control X-punches, in any positions, will be recognized when the underpunched digit is a zero. "Not recognized" means that the character position is replaced with a blank. This is the case for both input and output when standard FORTRAN READs and WRITEs are used.

A 12-punch is not recognized by the conversion routines with FORTRAN when the underpunched digit is a zero. Therefore, a plus zero (12-0 Hollerith) will be expressed as only a 0-punch. For this reason, plus fields should be left unzoned rather than 12-punched in the units position.

When the input routines supplied with this package are used, this problem does not exist. All zone punches are recognized and are treated properly.

CSP OVERLAPPED I/O

The CSP overlapped I/O routines have been provided to take advantage of the cycle-stealing capability of the 1130. Because many allow processing to be resumed before the I/O is finished, their use will increase the throughput rates of many programs.

The table below summarizes the overlap capabilities of the routines:

This device	is overlapped with this function
Card reader (1442 or 2501)	Conversion from card code to A1 format
Card punch	nothing (not overlapped)
Console keyboard	nothing (not overlapped)
Console printer	anything but the console keyboard
Printer (1132 or 1403)	anything

The CSP I/O routines also permit the reading and punching of the 11-0 and 12-0 punches, both of which must be avoided with standard FORTRAN I/O.

The use of the overlapped I/O portion of CSP is an "either/or" proposition. For nondisk I/O, the programmer must choose either the CSP overlapped routines or the standard FORTRAN routines. The two systems cannot be intermixed within the same program. Note the emphasis on nondisk. This exclusion does not apply to disk I/O, which may be used regardless which of the two systems is selected.

Use of the overlapped I/O routines also excludes the employment of the TRACE feature of FORTRAN, since it uses portions of the FORTRAN package for output.

The following routines are included in the CSP I/O group:

READ	PRINT	TYPER
PUNCH	SKIP	KEYBD
R2501	P1403	STACK
P1442	S1403	

If any of these routines are used, standard FORTRAN READ and WRITE commands may not appear in the same program.

When using Version 1 of the 1130 Disk Monitor System, the programmer must place the statement

```
CALL IOND
```

before any STOP or PAUSE statement. This will ensure that all pending I/O interrupts have been serviced before the CPU stops or pauses. IOND should not be called if Version 2 of the Monitor is in use.

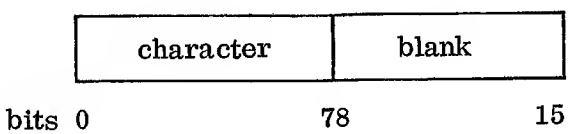
P1403, S1403, P1442, and R2501 use parts of the subroutine library supplied with Version 2 of the 1130 Disk Monitor System. If they are to be used with a Version 1 Monitor, the Version 2 subroutines must be loaded onto the Version 1 disk. See the detailed descriptions of P1403, S1403, P1442, and R2501 for more particulars.

DATA FORMATS USED

Although most of the CSP routines are oriented toward use of the A1 format, several new formats have been introduced. In addition, several of the standard formats must be considered in a different light.

A1 FORMAT

A1 format consists of one character per 16-bit word, left-justified:

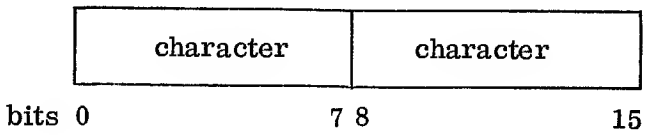


The right-hand eight bits should always contain the blank character, which is 01000000 in binary. This blank will always be inserted by the CSP routines and the standard FORTRAN A1 format.

The sign of an A1 field is assumed to be carried as an 11- or 12-punch over the rightmost character. An 11-punch is taken to signify a negative field; a 12-punch (or no-zone punch) signifies a positive field.

A2 FORMAT

A2 format consists of two characters per word:



A3 FORMAT

Although A3 format exists in standard FORTRAN terminology, its use in this manual has a different connotation. Here, A3 format means that one word contains three characters.

This can be done only by using a unique coding scheme. The user supplies a table of 40 characters. Then, the A1A3 and A3A1 subroutines may be used to translate from A1 to A3 format and vice versa.

The A3 format cannot be pictured graphically, since the three characters are combined as a single integer or binary number.

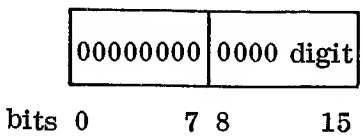
The A3 format permits highly efficient packing of alphabetic data and may be used to save considerable space on the disk.

Note, however, that only 40 characters may be used. This may not be enough for some applications. For example, if the characters chosen were A through Z, 0 through 9, the blank, comma, period, and dash, 40 would probably be ample for a name and address file. It would not be sufficient for a product description file that also required slashes, dollar signs, etc.

D1 FORMAT

D1 format consists of one digit per word, right-justified. Because the decimal arithmetic routines operate on data in this format, D1 format is also called decimal format.

D1 format is as follows:



A decimal field is stored in an array in D1 format. The sign of the field will be carried with the rightmost digit. For example, the six-digit field 001968 could be placed in the 12th through 17th position in the NUMBR array:

```
NUMBR(12) = 0
NUMBR(13) = 0
NUMBR(14) = 1
NUMBR(15) = 9
NUMBR(16) = 6
NUMBR(17) = 8
```

The same field, if it were negative, would be written as $001968\overline{8}$, and the sign would be reflected in the rightmost digit:

```
NUMBR(12) = 0
NUMBR(13) = 0
NUMBR(14) = 1
NUMBR(15) = 9
NUMBR(16) = 6
NUMBR(17)=-9
```

Note that NUMBR (17) is -9 rather than -8; this must be done because the 1130 cannot represent a negative zero. The following scheme is used with negative numbers:

If the sign of the field is negative and the rightmost digit is a	The rightmost D1 digit will be carried as a
0	-1
1	-2
2	-3
3	-4
4	-5
5	-6
6	-7
7	-8
8	-9
9	-10

Usually, this need not concern the programmer, since the A1DEC and DECA1 routines will automatically implement the special coding of negative fields. Setting up negative constants, though, must be handled properly by the programmer.

D4 FORMAT

D4 format consists in general of four decimal digits per word, with each digit occupying four bits of the word. However, since the sign digit (the rightmost one) carries the sign, it is handled separately, and is placed by itself in the last word of the D4 field. This is best illustrated by showing several examples:

The five-digit number +12345

first word				second word			
1	2	3	4				+5
0001	0010	0011	0100	0000	0000	0000	0101

The six-digit number +123456

first word				second word				third word			
1	2	3	4	5	F	F	F				+6
0001	0010	0011	0100	0101	1111	1111	1111	0000	0000	0000	0110

The seven-digit number +1234567

first word				second word				third word			
1	2	3	4	5	6	F	F				+7
0001	0010	0011	0100	0101	0110	1111	1111	0000	0000	0000	0111

The filler consists of four 1 bits, the hexadecimal F. A more detailed description of D4 format may be found with the description of the DPACK routine.

FORMAT REQUIREMENTS

The requirements for each subroutine are as follows:

Subroutine	Format of Data before Processing	Format of Data after Processing
ADD	D1 format	D1 format
A1A3	A1 format	A3 format
A1DEC	A1 format	D1 format
A3A1	A3 format	A1 format
CARRY	D1 format	D1 format
DECA1	D1 format	A1 format
DIV	D1 format	D1 format
DPACK	D1 format	D4 format
DUNPK	D4 format	D1 format
EDIT	A1 format	A1 format
FILL	Any integer (A1, A2, D1, etc.)	Same as FILL character
GET	A1 format	Real variable (extended precision)
ICOMP	D1 format	Greater than, equal to, or less than zero
IOND	None	None
KEYBD	A1 format	A1 format
MOVE	Any integer (A1, A2, D1, etc.)	Same as before MOVE
MPY	D1 format	D1 format
NCOMP	A1 format	Greater than, equal to, or less than zero
NSIGN	D1 format	Integer variable
NZONE	A1 format	Integer variable
PACK	A1 format	A2 format
PRINT	A1 format	A1 format
PUNCH	A1 format	A1 format
PUT	Real variable (extended precision)	A1 format
P1403	A1 format	A1 format
P1442	A1 format	A1 format
READ	A1 format	A1 format
R2501	A1 format	A1 format
SKIP	Decimal constant	None
STACK	None	None
SUB	D1 format	D1 format
S1403	Decimal constant	None
TYPER	A1 format	A1 format
UNPAC	A2 format	A1 format
WHOLE	Real variable (any precision)	Real variable (any precision)

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

DETAILED DESCRIPTIONS

This section gives the general format and a description of each routine. Each description contains format, function, parameter description, detailed description, example, errors, and remarks. The function describes the capabilities of the routine. The parameter description explains in detail how the parameters, variables, and constants should be set up. The detailed description tells exactly what the subroutine does and how it should be used. Examples are given as an aid to the programmer. Certain specification and input errors may occur when using the package, and these are explained. The remarks section describes some peculiarities of the routine. Further information may be obtained from the flowcharts and listings.

Format: CALL ADD(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Sums two arbitrary-length decimal data fields, placing the result in the second data field.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array which is added, the addend. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit to be added (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit to be added (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the augend, the array which is added to. It will contain the result in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of KCARD (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of KCARD (the right-hand end of a field).

NER - An integer variable. Upon completion of the subroutine, this variable indicates whether arithmetic overflow occurred.

Detailed description: The corresponding digits, by place value, of JCARD and KCARD, are summed and placed back in KCARD. This operation is from left to right, with both fields being right-adjusted. Next, all carries are set in order. If overflow occurred, it is indicated by NER being equal to KLAST. NER must be initialized and reset by the user. More detailed information may be found in the ADD flowchart and listing.

Example: DIMENSION IGRND(12),ITEM(6)

N=0

CALL ADD(ITEM,1,6,IGRND,1,12,N)

Before:

IGRND 000713665203 ITEM 102342

Position 1 5 10 Position 1 5

N=0

After:

IGRND 000713767545 ITEM is unchanged.

Position 1 5 10

N=0

The numeric data field ITEM, in decimal format, is ADDed to the numeric data field IGRND, also in decimal format. Note that the fields are both right-justified. The error indicator, N, is the same, since there is no overflow out of the high-order digit (left-hand end) of the IGRND field.

Errors: If the KCARD field is not large enough to contain the sum, that is, if there is a carry out of the high-order digit, the error indicator, NER, will be set equal to KLAST, and the KCARD field will be filled with 9s.

If the JCARD field is longer than the KCARD field, nothing will be done and the error indicator will be equal to KLAST.

Remarks: Conversion from EBCDIC to decimal is necessary before using this subroutine. This may be accomplished with the A1DEC subroutine.

The length of the JCARD and KCARD fields is arbitrary, up to the maximum space available.

Note that the error indicator is not reset by this subroutine. It is the responsibility of the user to initialize, test, and reset the error indicator.

Format: CALL A1A3(JCARD,J,JLAST,KCARD,K,ICHAR)

Function: To convert from A1 format (one character per word) to A3 format (three characters per word).

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the field to be converted. Originally, this field must be in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be converted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable. This is the position of the last character of JCARD to be converted (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is converted, in A3 format, three characters per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the converted characters (the left-hand end of a field).

ICHAR - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains a table used in the conversion.

Detailed description: Three characters in A1 format are taken, one at a time, from the JCARD array. The relative position of each character is found in the table ICHAR. Then these three relative positions are used to form an A3 integer as follows:

A3 INTEGER=(N1-20)*1600+(N2*40)+N3

where N1 is the relative position of the first character in the ICHAR array, etc. The A3 integer is then placed in the KCARD array, and the next group of three A1 characters is packed, and so on. Note that the relative position runs from 0 to 39, not 1 to 40.

Example: Set up ICHAR as follows:

```
      DIMENSION    ICHAR(40)
      READ(2,1) ICHAR
    1 FORMAT (40A1)
```

or

```
      DIMENSION ICHAR(40)
      CALL READ(ICHAR,1,40,N)
```

The card to be read is:

Content	E	T	A	O	I	N	b	S	H	R	D	L	U	C	M	F	W	Y	P	0	1	2	3	4	5	6	7	8	9	V	B	G	K	Q	J	X	Z	,	.	&
Card column	1				5				10				15				20			25		30			35			40											
Relative position	0				4				9				14				19			24		29			34			39											

It is the user's responsibility to create the ICHAR array. It must always contain 40 characters.

A1A3 may be used as follows:

```
      DIMENSION JCARD(21),  KCARD(10),ICHAR(40)
      CALL A1A3(JCARD,1,21,KCARD,1,ICHAR)
```

Before:

JCARD	C	U	S	T	O	M	E	R		N	A	M	E		I	S		H	E	R	E																		
Position	1				5					10					15				20																						

KCARD	0	1	2	3	4	5	6	7	8	9
Position	1				5					10

ICHAR is as above.

After:

JCARD is the same.
ICHAR is the same.

KCARD	-10713	-30266	-31634	-23906	-31756	-20552	-31640	7	8	9	
Position	1	2	3	4	5	6	7	8	9	10	
Represents	CUS	TOM	ER6	NAM	E6I	S6H	ERE				

The large negative numbers at each of the first seven positions reflect A3 integers (three A1 characters).

Errors: If a character does not appear in ICHAR, and does appear in JCARD, it will be coded as a blank.

Remarks: It is the user's responsibility to create the ICHAR array. It must always contain 40 characters. The arrangement shown in the example is, in general, the best, since the characters appear in the order of their most frequent occurrence, and this arrangement includes those characters (A-Z, 0-9, blank, comma, period, and ampersand) commonly found in alphabetic files (names and addresses, etc.). The user may, however, place any 40 characters in the ICHAR array, in any order.

If the field to be compressed consists primarily of numbers, for example, they should be placed first in the ICHAR array.

Note that the A3 format discussed here is a special one and is not the same as the FORTRAN A3 format.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

A1DEC

Format: CALL A1DEC(JCARD,J,JLAST,NER)

Function: Converts a field from A1 format, one digit per word, to decimal format, right-justified, one digit per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the name of the field that will be converted. Originally, this field must be in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be converted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be converted (the right-hand end of a field).

NER - An integer variable. This variable will be equal to the position of the last invalid (nonnumeric or nonblank) character encountered, except for the JLAST position, which may contain a sign.

Detailed description: The subroutine operates from left to right. Each character is checked for validity (digit or blank). Blanks are changed to zeros. If a character is invalid, the error indicator, NER, is set equal to the position of the character. If the character is valid, it is converted to decimal format and right-justified using the formula

$$\text{Decimal digit} = (\text{character}+4032)/256$$

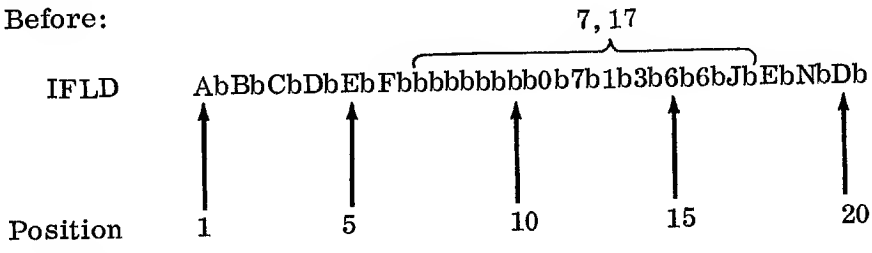
When all characters have been converted, the decimal field is signed. More detailed information may be found in the A1DEC flowchart and listing.

Example: DIMENSION IFLD(20)

N=0

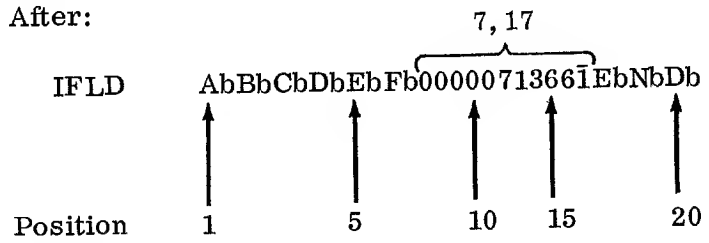
CALL A1DEC(IFLD,7,17,N)

Before:



N=0

After:



N=0

Before execution, the field is shown in A1 format, the character followed by a blank. Therefore, the field to be converted is

bbbb071366J

After execution, the field has been converted, as is evident. There were no invalid characters in the field, since N is the same.

Errors: If an invalid character (nonnumeric or nonblank) is encountered, the error indicator is set equal to the position of that character, and processing of the field continues.

Remarks: When the error indicator has been set, the character indicated is the last invalid character. There may be other invalid characters in the field, occurring to the left of the character noted.

Zone punches are used, at times, to indicate conditions (switches). These zones can be removed with the NZONE subroutine. Following is an error routine to correct errors of this type:

```
                Main Line
                    .
                    .
                    .
1       CALL A1DEC(IFLD,J,JLAST,N)
        IF(N) 2,2,3
2       Continue Main Line
                    .
                    .
                    .
3          Error Routine
        CALL NZONE(IFLD,N,4,N1)
        N1=0
        CALL A1DEC(IFLD,N,N,N1)
        IF(N1) 5,5,4
4       STOP 999
5       CALL DECA1(IFLD,J,JLAST,N)
        N=0
        GO TO 1
```

When an error of this type occurs, N will be greater than zero. Control would go to statement 3. Using the NZONE routine, the zone is removed (if not a special character). The invalid character is now converted with the A1DEC routine. If the character is still invalid, control goes to statement 4 and the program will STOP. If the character is now valid, it has been converted and control goes to statement 5. However, there may have been other invalid characters. Therefore, at statement 5 the field is converted back to A1 format and control returns to statement 1, where the field is again converted from A1 format to decimal format. This process continues until a truly invalid character (special character) is encountered, or until the field is converted with no errors.

Note that the error indicator is not reset by this subroutine. It is the responsibility of the user to initialize and reset the error indicator.

Format: CALL A3A1(JCARD,J,JLAST,KCARD,K,ICHAR)

Function: To convert from A3 format (three characters per word) as created by the A1A3 subroutine to A1 format (one character per word).

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the field to be converted. Originally, this field must be in A3 format, three characters per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first element of JCARD to be converted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable. This is the position of the last element of JCARD to be converted (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is converted, in A1 format, one character per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the converted characters (the left-hand end of a field).

ICHAR - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains a table used in the conversion.

Detailed description: A3 integers are taken, one at a time, from the JCARD array. Each is decoded into the three numbers of which it is composed, as follows:

$$N1=\begin{cases}(\text{A3 INTEGER}/1600)+20 & \text{if the A3 integer is positive}\\ ((\text{A3 INTEGER}+32000)/1600) & \text{if the A3 integer is negative}\end{cases}$$

N2=(A3 INTEGER-(N1-20)*1600)/40

N3=A3 INTEGER-(N1-20)*1600-(N2*40)

The resulting integers, N1, N2, N3, are then used to locate their corresponding A1 characters in the ICHAR array. Each A1 character is then placed in the KCARD array.

Note that each element of JCARD requires three elements in KCARD.

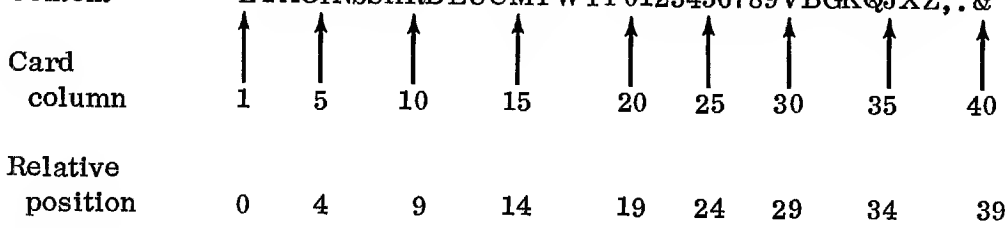
Example: Set up ICHAR as follows:

```
      DIMENSION     ICHAR(40)
      READ(2,1)  ICHAR
1     FORMAT (40A1)
```

or

```
      DIMENSION ICHAR(40)
      CALL READ(ICHAR,1,40,N)
```

The card to be read is:

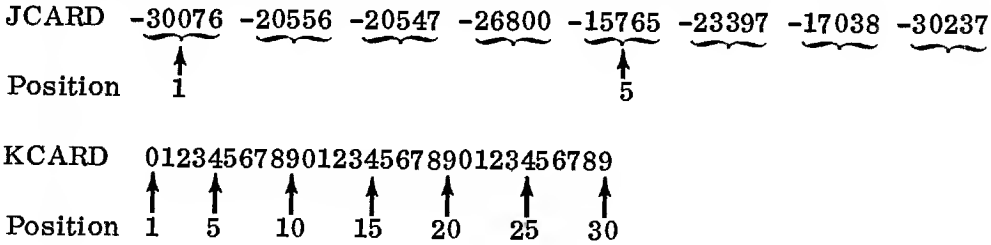


It is the user's responsibility to create the ICHAR array. It must always contain 40 characters.

A3A1 may be used as follows:

```
      DIMENSION JCARD(21),  KCARD(30),  ICHAR(40)

      CALL A3A1(JCARD,1,8,  KCARD,1,  ICHAR)
```

Before:

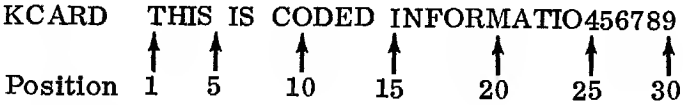


ICHAR is as above.

After:

JCARD is the same.

ICHAR is the same.



Errors: If JLAST is less than J, one element will be decoded into three characters.

Remarks: It is the user's responsibility to create the ICHAR array. It must always contain 40 characters. The arrangement shown in the example is, in general, the best, since it is in the order of the most frequent occurrence of the letters of the alphabet.

Note that the A3 format discussed here is a special one, and is not the same as the FORTRAN A3 format.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

CARRY

Format: CALL CARRY(JCARD,J,JLAST,KARRY)

Function: Resolve all carries within the specified field and indicate any high-order carry out of the field. This routine will not normally be called by the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the field that will be interrogated for carries. The data must be in decimal format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of JCARD (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD (the right-hand end of a field).

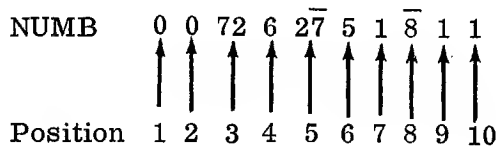
KARRY - An integer variable. This variable will contain any carry out of the high-order position of the JCARD field. If there is no carry, KARRY will be set to zero.

Detailed description: The routine operates from right to left, examining the low-order digit first. The digit being examined is divided by ten. Since only integers are used, the quotient of this division is the carry in that digit. Ten times the carry is subtracted from the digit. If the digit is now negative, ten is added to the digit and one is subtracted from the carry. At this point, or if the resultant digit was positive, the next digit to the left is examined. First, the carry from the previous digit is added to this digit. Then the process for the first digit, starting with division by ten, is carried out. When all digits have been examined, from JCARD(JLAST) to JCARD(J) inclusive, the final carry is set and the routine terminates. More detailed information may be found in the CARRY flowchart and listing.

Example: DIMENSION NUMB(10)

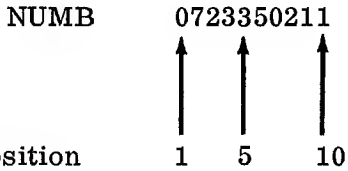
CALL CARRY(NUMB,1,10,N)

Before:



N=22

After:



N=0

After an arithmetic operation the condition of the NUMB field is as shown at "Before". The third, fifth and eighth positions appear as shown, because multiple arithmetic operations have generated them. The object of the CARRY routine is to resolve this type of problem.

Notice that a 1 has been borrowed from the seventh position to resolve the -8 condition. Similarly, a 3 has been borrowed from the fourth position, and the 7 from 72 has gone into the second position.

Errors: None

Remarks: This routine is used by the other routines in this package as a service routine. In general, the user need not call this routine, since all carries are resolved by the arithmetic routines themselves (ADD, SUB, MPY, DIV).

DECA1

Format: CALL DECA1(JCARD,J,JLAST,NER)

Function: Converts a field from decimal format, right-justified, one digit per word, to A1 format, one character per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the name of the field that will be converted. Originally, this field must be in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of JCARD to be converted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be converted (the right-hand end of a field).

NER - An integer variable. This variable will be equal to the position of the last digit of JCARD which was negative or greater than 9, except for the JLAST position, which can be negative (sign).

Detailed description: The subroutine operates from left to right. First the sign is determined. Then each digit, starting with JCARD(J), is converted to A1 format using the formula

Character = 256 * (decimal digit) - 4032

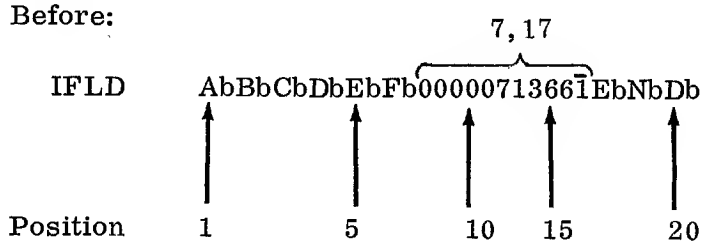
When all digits have been converted, the field is signed. More detailed information may be found in the DECA1 flowchart and listing.

Example: DIMENSION IFLD(20)

N=0

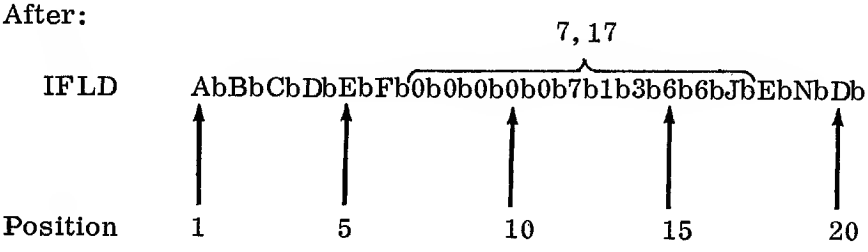
CALL DECA1(IFLD,7,17,N)

Before:



N=0

After:



N=0

Before execution the field is shown in decimal format. The field to be converted is

00000713661

After execution, the field has been converted to A1 format, as is evident, the character followed by a blank. There were no invalid digits in the field, since N is the same.

Errors: If an invalid digit (not 0 to 9, inclusive) is encountered, the error indicator is set equal to the position of that character, and processing of the field continues.

Remarks: When the error indicator indicates an error, the digit indicated is the last invalid digit. There may be other invalid digits in the field, occurring to the left of the digit noted.

These errors should not occur, since the arithmetic routines (ADD, SUB, MPY, and DIV) will resolve carries. However, if this does happen, the user's program should indicate (possibly by STOPing) that this has occurred.

Note that the error indicator is not reset by this subroutine. It is the responsibility of the user to initialize and reset the error indicator.

DIV

Format: CALL DIV(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Divides one arbitrary-length decimal data field by another, placing the quotient and remainder in the dividend.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array is the divisor. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of the divisor (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit of the divisor (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array, the dividend, will contain the quotient and the remainder, extended to the left, in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of the dividend (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last digit of the dividend (the right-hand end of a field). This is also the position of the last digit of the remainder.

NER - An integer variable. Upon completion of the subroutine, this variable indicates whether division by zero was attempted, or whether the KCARD field is not long enough.

Detailed description: First the signs are cleared from both fields and saved. Then the KCARD field is extended to the left the length of the JCARD field (JLAST-J+1), and filled with zeros. If the KCARD field will be extended below KCARD(1), NER will be set equal to KLAST and the routine will be terminated. Next, the JCARD field is scanned to find the high-order significant digit. If no digit is found, the error indicator NER is set to KLAST, and the result is the same as the input. When a digit is found, the division begins. It is done by the method of trial divisors:

1. The high-order digit of the divisor is used as the trial divisor.

2. The trial divisor is divided into the next high-order digit of the dividend to generate a digit of the quotient.

3. The digit of the quotient is multiplied by the trial divisor.

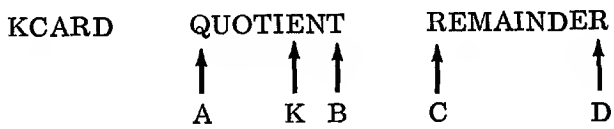
4. This product is subtracted from the corresponding number of digits in the high-order portion of the dividend.

5. As long as the result is positive, the quotient digit is the next digit in the quotient. A return is made to step 2.

6. When the result is negative, the product from step 3 is added back to the dividend, 1 is subtracted from the quotient digit, and the new quotient digit is placed in the quotient as the next digit. Finally, the signs are generated for the quotient and remainder and the sign is replaced on the divisor.

The quotient will be located in the KCARD field. The subscript of the first digit of the quotient will be K-(JLAST-J+1), and the subscript of the last digit of the quotient will be KLAST-(JLAST-J+1).

The remainder will also be located in the KCARD field. The subscript of the first digit of the remainder will be KLAST-JLAST+J, and the subscript of the last digit of the remainder will be KLAST.



A is the position whose subscript is K-(JLAST-J+1).
K is the first position of the dividend, defined earlier.
B is the position whose subscript is KLAST-(JLAST-J+1).
C is the position whose subscript is KLAST-(JLAST-J).
D is the position whose subscript is KLAST.

More detailed information may be found in the DIV flowchart and listing.

Example:

```
DIMENSION IDVSR(5),IDVND(15)
N=0
CALL DIV(IDVSR,1,5,IDVND,6,15,N)
```

Before:

IDVSR 00982 — Position 1 5

IDVND ABCDE0007136673 — Position 1 5 10 15

N=0

After:

IDVSR is unchanged.

IDVND 000000726700479 — Position 1 5 10 15

N=0

The numeric data field IDVND has been divided by the numeric data field IDVSR, the quotient and remainder being placed in IDVND. Note that the IDVND field has been extended to the left the length of the IDVSR field, five positions.

Errors: If division by zero is attempted, the only action is that KCARD is extended and filled with zeros. The error indicator indicates that division by zero was attempted (NER=KLAST).

If there is not enough room to extend the KCARD field to the left, NER will again be set equal to KLAST, and the routine will terminate. None of the fields involved will be modified.

Remarks: Conversion from EBCDIC to decimal is necessary before using this subroutine. This may be accomplished with the A1DEC subroutine.

The length of the JCARD and KCARD fields is arbitrary, up to the maximum space available.

The arithmetic performed is decimal arithmetic, using whole numbers only. No decimal point alignment is allowed. For this reason numbers should have an assumed decimal point at the right-hand end.

Space must always be provided in the KCARD field for expansion. The first position of the dividend, K, must be at least JLAST-J+1 positions from the beginning of KCARD. For example, if JCARD is seven positions, 1 through 7, the dividend in KCARD must start at least seven positions (7-1+1=7) from the beginning of KCARD. This would have K equal to 8.

DPACK

Format: CALL DPACK(JCARD, J, JLAST, KCARD, K)

Function: Information in D1 format, one digit per word, is packed into D4 format, four digits per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the data to be packed, in D1 format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be packed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable greater than J. This is the position of the last character of JCARD to be packed (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is packed, in D4 format, four digits per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the packed characters (the left-hand end of a field).

Detailed description: Initially, the field to be packed (the JCARD array) is in D1 format. This consists of one digit per word, right-justified (occupying the rightmost four bits of the word). The sign of the field is carried with the rightmost or low-order digit.

The operation of the DPACK subroutine is as follows: Starting at JCARD(J), and working from left to right, each four-bit digit of the JCARD array is placed into four bits of the KCARD array, four to the word, starting at KCARD(K). When JCARD(JLAST) is encountered, it is assumed to be the last D1 digit, and to carry the sign of the field. The DPACK routine then places JCARD(JLAST), unpacked, in its entirety, into KCARD((JLAST-J+7)/4), the last position in the KCARD array.

Any unused space in the preceding KCARD word is then filled with 1 bits. This bit arrangement or format will be called D4 format.

For example, suppose a seven-position JCARD array is to be packed, and it contains 1, 2, 3, 4, 5, 6, 7:

JCARD(1) = 1
JCARD(2) = 2
JCARD(3) = 3
JCARD(4) = 4

JCARD(5) = 5
JCARD(6) = 6
JCARD(7) = 7

JCARD(1) through JCARD(4) will be placed in KCARD(1) as 0001 0010 0011 0100.

JCARD(5) and JCARD(6) will be placed in KCARD(2) as 0101 0110 0000 0000.

JCARD(7) will be placed, without conversion, in KCARD(3) as 0000 0000 0000 0111.

Then the two unused four-bit areas in KCARD(2) will be filled with 1's as 0101 0110 1111 1111.

More detailed information may be found in the DPACK/DUNPK flowchart and listing.

The table below may be used to determine the number of words required for a field after it is packed. For example, a twelve-digit decimal field will be packed into a four-word field:

- First word: 1st, 2nd, 3rd, and 4th digits
- Second word: 5th, 6th, 7th and 8th digits
- Third word: 9th, 10th, and 11th digits, plus four 1 bits (filler)
- Fourth word: 12th digit carrying the sign of the field.

Field Length		Field Length		Field Length	
Before Packing	After Packing	Before Packing	After Packing	Before Packing	After Packing
2	2	18	6	34	10
3	2	19	6	35	10
4	2	20	6	36	10
5	2	21	6	37	10
6	3	22	7	38	11
7	3	23	7	39	11
8	3	24	7	40	11
9	3	25	7	41	11
10	4	26	8	42	12
11	4	27	8	43	12
12	4	28	8	44	12
13	4	29	8	45	12
14	5	30	9	46	13
15	5	31	9	47	13
16	5	32	9	48	13
17	5	33	9	49	13

Example: DIMENSION IUNPK(26),IPAKD(26)

CALL DPACK(IUNPK, 1, 10, IPAKD, 1)

Before:

IUNPK 123456789123

Position 1 5 10

IPAKD ABCDEFGHIJ

Position 1 5 10

After:

IUNPK is the same.

IPAKD 1234 5678 9FFF 0001 EFGHIJ

Position 1 2 3 4 5 6 10

Errors: None

Remarks: If JLAST is less than or equal to J, only one character of JCARD will be packed, and it will be treated as the sign. A multiple of four characters in JCARD will always be packed into KCARD. An equation for how much space is required, in elements, in KCARD is:

$$\text{Space in KCARD} = \frac{\text{JLAST-J+7}}{4}$$

This result is rounded down at all times.

DUNPK

Format: CALL DUNPK(JCARD, J, JLAST, KCARD, K)

Function: Information in D4 format, four digits per word, is unpacked into D1 format, one digit per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the data to be unpacked, in D4 format, four digits per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first element of JCARD to be unpacked (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable greater than J. This is the position of the last element of JCARD to be unpacked, (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is unpacked, in D1 format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the unpacked characters (the left-hand end of a field).

Detailed description: See the detailed description of DPACK for an explanation of the D1 and D4 formats.

The JCARD field, in packed (D4) format, will be unpacked (converted to D1 format) and placed in the KCARD field. Starting at JCARD(J), moving from left to right, each four-bit digit is placed in the rightmost four bits of a word in the KCARD array, starting at KCARD(K).

Filler bits (four 1's) are recognized as such and are ignored.

JCARD(JLAST), the last word to be converted, is not altered, but is moved to KCARD(KLAST). KLAST cannot be calculated exactly at this point, but KLAST-K+1 will be the same as JLAST-J+1 when the field was originally packed. In other words, field lengths will not be changed by a DPACK and subsequent DUNPK.

The maximum value of KLAST can be calculated as

$$4*(JLAST-J)+1$$

However, it may be one, two, or three fewer positions in length.

More detailed information may be found in the DPACK/DUNPK flowchart and listing.

Example:

```
DIMENSION IUNPK(26),IPAKD(26)

CALL DUNPK(IPAKD,1,3,IUNPK,1)
```

Before:

```
IPAKD     1234 5678 0003          IUNPK     FbIbLbLbbbIbNbbbTbHbIbSb
                  ↑    ↑    ↑                      ↑        ↑        ↑
                  1    2    3                      1        5        10
Position                                  Position
```

After:

IPAKD is the same.

```
IUNPK     123456783HbIbSb
                  ↑    ↑    ↑
                  1    5    10
Position
```

Errors: None

Remarks: If JLAST is less than or equal to J, only the first element of JCARD, JCARD(J) will be unpacked and it will be treated as the sign.

EDIT

Format: CALL EDIT(JCARD, J, JLAST, KCARD, K, KLAST)

Function: Edits data from one array into another array, which contains the edit mask.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the data to be edited, called the source field, one character per word, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be edited (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be edited (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which data is edited; it contains the edit mask before editing begins, stored one character per word, in A1 format, and is called the mask field.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of the edit mask (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than K. This is the position of the last character of the edit mask (the right-hand end of a field).

Detailed description: The following table gives the control characters for editing, the characters used to make up the mask, and their respective functions:

Control Character	Function
b (blank)	This character is replaced by a character from the source field.
0 (zero)	This character indicates zero suppression and is replaced by a character from the source field. The position of this character indicates the rightmost limit of zero suppression (see description of operation below). Blanks are inserted in the high-order nonsignificant positions of the field.

Control Character	Function
. (decimal point)	This character remains in the mask field where placed. However, if zero suppression is requested, it will be removed if it is to the left of the last character to be zero-suppressed.
, (comma)	This character remains in the mask field where placed. However, if zero suppression is requested, it will be removed if it is to the left of the last character to be zero-suppressed.
CR (credit)	These two characters can be placed in the two rightmost positions of the mask field. They are undisturbed if the source field is negative. (If the source field is positive, the characters C and R are blanked out.) In editing operations, a negative source field is indicated by an 11-zone over the rightmost character. Whether CR is blanked out or not, no data will be edited into these positions when CR is present, but rather into the edit characters to the left. The letters C and R may be used in the remainder of the edit mask, where they will be treated as normal alphabetic characters, without being subject to sign control. Only the R character is checked, so the C character may be any legal character, and it will be treated as described.
- (minus)	This character is handled similarly to CR in the rightmost position of the mask field.
* (asterisk)	This character operates the same as the 0 (zero) for zero suppression, except that asterisks rather than blanks are inserted in the high-order nonsignificant positions of the field, providing asterisk check protection.
$ (floating dollar sign)	This character has the same effect as the 0 (zero) for zero suppression, except that a $ is inserted to the left of the first significant character found, or to the left of the position that stopped the zero suppression.

The operation of the edit routine may be described in five steps:

1. Characters are placed in the mask field from the source field, moving from right to left. The characters 0 (zero), b (blank), * (asterisk) and $ (dollar sign) are replaced with characters from the source field. No other characters in the mask field are disturbed.

2. If all characters in the source field have not been placed in the mask field before the end of the mask field is encountered, the whole mask is set to asterisks and editing is terminated.

3. CR (credit) and - (minus) in the rightmost positions of the mask field are blanked if the source field is positive (does not have an 11-zone over the rightmost character).

4. The zero suppression scan starts at the left end of the mask field and proceeds left to right, replacing zeros (0), blanks (b's), decimal points (.), and commas (,). The last position replaced will occur where the zero suppression character was located, or one position to the left of where a significant character, not zero (0), blank (b), decimal point (.), or comma (,), occurs. If the zero suppression character was an asterisk (*), the replacement character is an asterisk. Otherwise, the replacement character is a b (blank).

5. If the zero suppression character was a dollar sign ($), a dollar sign is placed in the last replaced position in the zero suppression scan.

In order for the edit routine to work correctly and as described, five rules must be followed in creating the mask field:

1. There must be at least as many b's (blanks) in the mask field as characters in the source field.

2. If the mask field contains zero (0), asterisk (*), or dollar sign ($), zero suppression will be used and the first character in the mask field must be a b (blank).

3. The mask field must not contain more than one of the following, which may appear only once:

   0 (zero)

   * (asterisk)

   $ (dollar sign)

4. If the rightmost character in the mask field is an R, the next character to the left should be a C, in order to edit with CR (credit). Both characters will be blanked if the source field is positive. If the rightmost character in the mask field is - (minus), it will be blanked if the source field is positive.

5. All numeric, alphabetic, and special characters may be used in the mask field. All characters that do not have special meaning will be left in their original position in the mask field during the edit.

   More detailed information may be found in the EDIT flowchart and listing.

Example: There are three common methods for creating a mask field such as b,bb$.bbCR:

Method 1

```
      DIMENSION MASK(10)
1     FORMAT(10A1)
      IN=2
      READ(IN,1)MASK
```

Method 2

```
DIMENSION MASK(10)
MASK(1)=16448
MASK(2)=27456
MASK(3)=16448
MASK(4)=16448
MASK(5)=23360
MASK(6)=19264
MASK(7)=16448
MASK(8)=16448
MASK(9)=-15552
MASK(10)=-9920
```

Method 3

```
DIMENSION MASK(10)
DATA MASK/'b','','b','b','$','.','b','b','C','R'/
```

Method 1 creates the mask by reading it from a card. Method 2 creates the mask with FORTRAN arithmetic statements, setting each position of the mask to the desired character. It uses the decimal equivalents of the various EBCDIC codes, as listed in the APPENDIX. Method 3, using the DATA statement, is by far the shortest and simplest. Note that each character requires a word of core storage, regardless of the method employed.

The table of examples below illustrates how the EDIT routine works:

Source Field	Mask Field	Result
00123D	bb,bb$.bbCR	bbb$12.34bb
00123M	bb,bb$.bbCR	bbb$12.34CR
00123M	bb,bb$.bb-	bbb$12.34-
00123D	bb,bb$.bb-	bbb$12.34b
46426723	b,bbb,bb$.bbCR	b$464,267.23bb
00200P	b,bb*.bbCR	***20.07CR
082267139	bbb-bb-bbbb	082-26-7139
01234567	bbbb$.bbCR	**********
0AB1234	bbbbb$.bbCR	b$AB12.34bb
-12345	bb,bb$.bb-	$-,123.45b

Because the mask field is destroyed after each use, it is advisable to move the mask field to the output area and perform the edit function in the output area.

Errors: If the number of characters in the source field is greater than the number of blanks in the mask field, the mask field is filled with asterisks(*).

Format: CALL FILL(JCARD,J,JLAST,NCH)

Function: Fills an area with a specified character.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the area to be filled.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be filled (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be filled (the right-hand end of a field).

NCH - An integer constant, an integer expression, or an integer variable. This is the code for the fill character. The Appendix contains a list of those codes corresponding to the EBCDIC character set; however, NCH may be any integer.

Detailed description: The area of JCARD, starting with J and ending with JLAST, is filled with the character equivalent to the NCH code, one character per word. More detailed information may be found in the FILL flowchart and listing.

Example: CALL FILL (IPRNT,3,10,16448)

Fill the area IPRNT from positions 3 through 10 with blanks. In other words, clear the area.

IPRNT:

Before: A B C D E F G H I J K L M N O P Q R S b . . .

After: A B b b b b b b b b K L M N O P Q R S b . . .

Position 1 5 10 15 20

Errors: None.

GET

Format: GET (JCARD, J, JLAST, SHIFT)

Function: Extracts a data field from an array, and converts it to a real number. This is a function subprogram.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the data to be retrieved, stored one digit per word, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be retrieved (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be retrieved (the right-hand end of a field).

SHIFT - A real constant, a real expression, or a real variable. If decimal places are required, SHIFT is equal to 10^{-d}, d being the number of decimal places. When SHIFT is used as a scale factor, SHIFT is 10^{d}, d being the number of zeros. If a card contains 12345 and the value of SHIFT is 0.0001, the result will be 1.2345. The result will be 123450. if a value 10.0 is assigned to SHIFT.

Detailed description: Using the formula

BINARY DIGIT = (EBCDIC CODE + 4032) / 256

the real digits are retrieved. Each binary digit is shifted left and summed, resulting in a whole number decimal. The sum is multiplied by SHIFT to locate the decimal point. The result is then placed in the real variable GET. If there are blanks in the data field, they are treated as zeros. If a nonnumeric character, other than blank, appears in any position other than the low-order position, the variable containing the result is zero. If a special character, other than the - (minus), appears in the low-order position, the resulting variable is set to zero.

For input and for output the sign must be placed over the low-order position as an 11-punch for minus and a 12 or no overpunch for plus. If the low-order position is zero and the number is negative, the column must contain only an 11-punch. (The zero must not be punched when FORTRAN I/O is used.) If the low-order position is zero and the number is positive, the column must contain only the zero punch. (The 12 row must not be punched when FORTRAN I/O is used.)

More detailed information may be found in the GET flowchart and listing.

Example 1: DIMENSION INCRD(80)

B=GET(INCRD,1,5,0.001)

Before: INCRD 0123456b...

B = 0.0

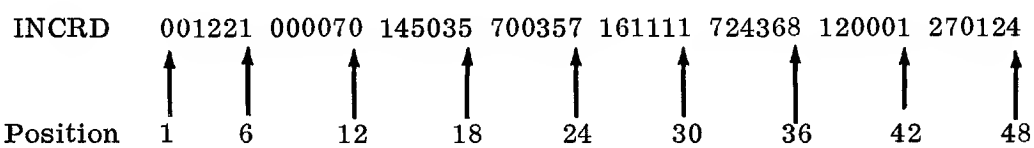
After: INCRD is the same.

B = 1.234 (Approximately, since a fraction is present)

Example 2:

A = GET (INCRD,1,6,1.0) + GET (INCRD,7,12,1.0)
+ GET (INCRD,13,18,1.0) + GET (INCRD,19,24,1.0)
+ GET (INCRD,25,30,1.0) + GET (INCRD,31,36,1.0)
+ GET (INCRD,37,42,1.0) + GET (INCRD,43,48,1.0)

Before:



A=0.0

After: INCRD is the same

A = 2122287. (Exactly, since no fractions were generated)

The above example sums the six-digit fields found in the first 48 columns of a card. Each data field has two decimal places. Any arithmetic operation can be performed with GET () as an operand.

Errors: If a nonnumeric character, other than blank, appears in a position other than the low-order position, the result is set to zero.

If a special character other than - (minus) appears in the low-order position, the result is set to zero.

Remarks: The GET routine is a function subprogram. As such, it is used in an arithmetic expression as shown in the example.

When using standard FORTRAN I/O, and the digit in the units position is a zero, a minus sign is shown as an 11-punch only; a plus is shown as a zero-punch only.

In most cases the value of SHIFT should be 1.0, placing the decimal point at the right-hand end of the number. (For dollars and cents calculations, the result of the GET would be in cents.) This will eliminate precision errors from the calculations. The decimal point may be replaced (moved to the left) with the EDIT routine for output.

If GET (or PUT) is used, the calling program must use extended precision.

Format: ICOMP (JCARD,J,JLAST,KCARD,K,KLAST)

Function: Two variable-length decimal format data fields are compared. The result is set to a negative number, zero, or a positive number. This is a function subprogram.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the first data field to be compared, one digit per word, in decimal format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be compared (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be compared (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the second data field to be compared, one digit per word, in decimal format. If the fields are unequal in length, the KCARD field must be the longer field.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of KCARD to be compared (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of KCARD to be compared (the right-hand end of a field).

Detailed description: Since the fields are assumed to be right-justified, the first operation is to examine the length of each field. If KCARD is longer than JCARD, the leading digits of KCARD are examined. If any one of them is greater than zero the result (ICOMP) is the opposite sign of KCARD. If they are all zero, or if the lengths are equal, corresponding digits are compared. The routine operates from left to right. The routine terminates when KCARD is longer than JCARD and a nonzero digit appears in the high-order of KCARD, when JCARD and KCARD do not match, or when all digits in JCARD and KCARD are equal. The following table shows the value of ICOMP, depending on the relation of the JCARD field to the KCARD field:

ICOMP	Relation
- (minus)	JCARD is less than KCARD
0 (zero)	JCARD is equal to KCARD
+ (plus)	JCARD is greater than KCARD

More detailed information may be found in the ICOMP flowchart and listing.

Example: DIMENSION ITOT(10),ICTL(10)

IF (ICOMP(ICTL,1,10,ITOT,1,10)) 1,2,1

The control total is compared to the total calculated. Control goes to statement 1 if the totals do not match (the calculated total is greater than or less than the control total). Control goes to statement 2 if the calculated total is equal to the control total. The fields compared are not changed.

ITOT	0007136673
ICTL	0007136688
ICOMP	after is positive.

Errors: No errors are detected. However, the JCARD field must not be longer than the KCARD field.

Remarks: ICOMP is a function subprogram and as such should be used in an arithmetic expression.

If JLAST is less than J, or KLAST is less than K, the result is unpredictable.

Format: CALL IOND

Function: Checks for I/O interrupts and loops until no I/O interrupts are pending.

This subroutine should not be used in conjunction with Version 2 of the 1130 Disk Monitor System. It is unneeded; besides, it may not operate correctly. It (IOND) is required only for programs operating under control of Version 1 of the Monitor.

Detailed description: The routine checks the Interrupt Service Subroutine Counter to see whether any I/O interrupts are pending. If the counter is not zero, the routine continues to check it until it becomes zero. Then the routine returns control to the user. More detailed information may be found in the IOND flowchart and listing.

```
Example:    CALL IOND

            PAUSE 777
```

The two statements shown will wait until all I/O interrupts have been serviced. Then the program will PAUSE. If an I/O interrupt is pending, and IOND is not used before a PAUSE, the program will not PAUSE.

Errors: None

Remarks: This statement must always be used before a STOP or PAUSE statement.

It may also be helpful in debugging programs. Sometimes, with more than one event going on at the same time (PRINTing and processing) during debugging, difficulties can be encountered. The user may not be able to easily find the cause of trouble. The use of IOND after each I/O statement will ensure that only one I/O operation is going on at any given time.

KEYBD

Format: CALL KEYBD(JCARD,J,JLAST)

Function: Reads characters from the keyboard.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array will contain the keyed information when reading is finished. The information will be in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first word of JCARD into which a character will be keyed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last word of JCARD into which a character will be keyed (the right-hand end of a field).

Detailed description: The keyboard is read and the information being read is printed on the console printer. When the specified number of characters have been read, or when EOF is encountered, the reading terminates. The characters read are converted from keyboard codes to EBCDIC and placed in A1 format, one character per word. Control is now returned to the user. More detailed information may be found in the TYPER/KEYBD flowchart and listing.

Example:

```
DIMENSION INPUT(30)

CALL KEYBD(INPUT,1,27)
```

Before:

```
INPUT     ABCDEFGHIJKLMNOPQRSTUVWXYZ0123
          ↑    ↑    ↑    ↑    ↑    ↑    ↑
Position  1    5    10   15   20   25   30
```

After:

```
INPUT     THE CUSTOMER NAME GOES HERE123
          ↑    ↑    ↑    ↑    ↑    ↑    ↑
Position  1    5    10   15   20   25   30
```

The array INPUT, from INPUT(1) to INPUT(27), has been filled with information read from the keyboard.

Errors: The following WAITs may occur:

WAIT (loc)	Accumulator (hex)	Action
41	2xx0	Ready the keyboard.
41	2xx1	Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.

Only 60 characters at a time may be read from the keyboard.

If more than 60 characters are specified (JLAST-J+1 is greater than 60), only 60 characters will be read.

Remarks: The characters asterisked in Appendix D of IBM 1130 Subroutine Library (C26-5929) will be entered into core storage and printed. All other characters will be entered into core storage but will not be printed.

If this subroutine is used, all other I/O must use commercial routines.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE ←
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

MOVE

Format: CALL MOVE(JCARD,J,JLAST,KCARD,K)

Function: Moves data from one array to another array.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array from which data is moved. The data may be stored in JCARD in any format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be moved (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be moved (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array to which data is moved, one character per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of KCARD to which data will be moved (the left-hand end of a field).

Detailed description: Characters are moved, left to right, from the sending field, JCARD, starting with JCARD(J) and ending with JCARD(JLAST), to the receiving field KCARD, starting with KCARD(K). More detailed information may be found in the MOVE flowchart and listing.

Example: DIMENSION INPUT(80),IOUT(120)

L=20

K=14

CALL MOVE(INPUT,6,L,IOUT,K)

Before:

INPUT

bbbb12ABC45ZYXPQR999Ab...

Position 1 5 10 15 20

IOUT

bbbbbb1bb77b6ABCDEFGHIJKLMNOPb...

Position 1 5 10 15 20 25 30

After:

INPUT is the same.

IOUT

bbbbbb1bb77b62ABC45ZYXPQR999Pb...

Position 1 5 10 15 20 25 30

The field in the array INPUT, starting at INPUT(6) and ending at INPUT(20), is moved to the field in the array IOUT, starting at IOUT(14). A total of 15 characters are moved.

Errors: None

MPY

Format: CALL MPY(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Multiplies two arbitrary-length decimal data fields, placing the product in the second data field.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array is the multiplier. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit that will multiply (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit to multiply (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array, the multiplicand, will contain the product, extended to the left, in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of the multiplicand (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of the product and the multiplicand (the right-hand end of a field).

NER - An integer variable. This variable will indicate whether the KCARD field is not long enough.

Detailed description: First the signs are cleared from both fields and saved. Then the KCARD field is extended to the left the length of the JCARD field (JLAST-J+1) and filled with zeros. If the KCARD field will be extended below KCARD (1), NER will be set equal to KLAST and the routine will be terminated. Next, the JCARD field is scanned to find the high-order significant digit. If no digit is found, the result is set to zero. When a digit is found, the actual multiplication begins. The significant digits in the JCARD field are multiplied by the digits in the KCARD field, one at a time, starting with KCARD(K) and ending with KCARD(KLAST). The preliminary results are summed, shifting after each preliminary multiplication to give the correct place value to the preliminary results. Finally, the correct sign is generated for the result, in KCARD, and the sign of JCARD is restored. More detailed information may be found in the MPY flowchart and listing.

Example: DIMENSION MPLR(5),MCAND(15)

N=0

CALL MPY(MPLR,1,5,MCAND,6,15,N)

Before:

MPLR 00982

Position 1 5

MCAND ABCDE0007136673

Position 1 5 10 15

N=0

After:

MPLR is unchanged.

N=0

MCAND 000007008212886

Position 1 5 10 15

The numeric data fields MPLR and MCAND are multiplied, the result being placed in MCAND. Note that the MCAND field has been extended to the left the length of the MPLR field, five positions, and that N has not been changed.

Errors: If there is not enough room to extend the KCARD field to the left, NER will be set equal to KLAST, and the routine will terminate.

Remarks: Conversion from EBCDIC to decimal is necessary before using this subroutine. This may be accomplished with the A1DEC subroutine. The length of the JCARD and KCARD fields is arbitrary, up to the maximum space available.

The arithmetic performed is decimal arithmetic, using whole numbers only.

Space must always be provided in the KCARD field for expansion. The first position of the multiplicand, K, must be at least JLAST-J+1 positions from the beginning of KCARD. For example, if JCARD is 7 positions, 1 through 7, then the multiplicand, in KCARD, must start at least seven positions (7-1+1=7) from the beginning of KCARD. This would have K equal to 8.

The product, located in the KCARD field, will begin at position K-(JLAST-J+1) of KCARD, and end at position KLAST of KCARD.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP ←
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

NCOMP

Format: NCOMP(JCARD,J,JLAST,KCARD,K)

Function: Two variable-length data fields are compared, and the result is set to a negative number, zero, or a positive number. This is a function subprogram.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the first data field to be compared, one character per word, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be compared (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be compared (the right-hand end of a field).

KCARD - The name of a one-dimensional, integer array defined in a DIMENSION statement. This array contains the second data field to be compared, one character per word, in A1 format.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of KCARD to be compared (the left-hand end of a field).

Detailed description: Corresponding characters of JCARD and KCARD are compared logically, starting with JCARD(J) and KCARD(K). The routine operates from left to right. The routine terminates when JCARD and KCARD do not match, or when the character at JCARD(JLAST) has been compared. The following table shows the value of NCOMP, depending on the relation of the JCARD field to the KCARD field:

NCOMP	Relation
- (minus)	JCARD is less than KCARD
0 (zero)	JCARD is equal to KCARD
+ (plus)	JCARD is greater than KCARD

More detailed information may be found in the NCOMP flowchart and listing.

Example: DIMENSION IN(80),MASTR(80)

IF (NCOMP(IN,1,20,MASTR,1))1,2,3

The field on the input card starting in column 1 and ending in column 20 is compared with the master field. Control goes to statement 1 if the input card is less than the master card. Control goes to statement 2 if the input card equals the master card. Control goes to statement 3 if the input card is greater than the master card. The fields compared are not changed.

IN	1234567bbbbbbbABCDEF
MASTR	1234567bbbbbbbABCDEF
NCOMP after is zero	

Errors: None

Remarks: The collating sequence in ascending order is as follows:

A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,0,1,2,3,4,5,6,7,8,9,

blank,.,<,(,+,&,$,*,),-,/,,,%,#,@,',=

The compare operation is terminated by the last character of the first data field, the data field at JCARD, or by an unequal comparison. NCOMP is a function subprogram and as such should be used in an arithmetic statement.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN ←
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

NSIGN

Format: CALL NSIGN(JCARD,J,NEWS,NOLDS)

Function: Interrogate the sign and return with a code as to what the sign is. Also, modify the sign as specified.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the digit to be interrogated or modified, in decimal (D1) format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the digit to be interrogated or modified.

NEWS - An integer constant, an integer expression, or an integer variable. This is the code specifying the desired modification of the sign.

NOLDS - An integer variable. Upon completion of the routine, this variable contains the code specifying what the sign was.

Detailed description: The sign is retrieved and NOLDS is set as in the table below:

NOLDS is	When the sign was
+1	positive
-1	negative

Then a new sign is inserted, specified by NEWS, as shown in the table below:

NEWS	Sign
+1	positive
0	opposite of old sign
-1	negative
NOLDS	no change

More detailed information may be found in the NSIGN flowchart and listing.

Example:	DIMENSION INUMB(9) CALL NSIGN(INUMB,9,0,N)
Before:	N=0, INUMB(9)=7
After:	N=1, INUMB(9)= -7

Errors: None

Remarks: The digit processed must be in decimal (D1) format. If it is not, the results are meaningless.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE ←
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

NZONE

Format: CALL NZONE(JCARD,J,NEWZ,NOLDZ)

Function: Interrogate the zone and return with a code as to what the zone is. Also, modify the zone as specified.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the character to be interrogated or modified, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the character in JCARD to be interrogated or modified.

NEWZ - An integer constant, an integer expression, or an integer variable. This is the code specifying the modification of the zone.

NOLDZ - An integer variable. This variable contains the code specifying what the zone was.

Detailed description: The zone is retrieved and NOLDZ is set as in the table below:

NOLDZ is	When the character was
1	A-I
2	J-R
3	S-Z
4	0-9
more than 4	special

Then a new zone is inserted, specified by NEWZ, as shown in the table below:

NEWZ	Character
1	12 zone
2	11 zone
3	0 zone
4	no zone
more than 4	no change

When a special character is the original character, the zone will not be changed. More detailed information may be found in the NZONE flowchart and listing.

Example:	DIMENSION IN(80) CALL NZONE(IN,1,2,J)
Before:	J = 0 IN(1) = a B (a 12, 2 punch)
After:	J = 1 IN(1) = a K (an 11, 2 punch)

Errors: None

Remarks: The minus sign or dash (-, an 11-punch) is treated as if it were a negative zero, not as a special character. This is the only exception.

The only modification performed on an input minus sign is that it may be transformed to a digit zero with no zone (a positive zero).

PACK

Format: CALL PACK(JCARD,J,JLAST,KCARD,K)

Function: Information in A1 format, one character per word, is PACKed into A2 format, two characters per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the input array, containing the data in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be PACKed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than J. This is the position of the last character of JCARD to be PACKed (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is PACKed, in A2 format, two characters per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the PACKed characters (the left-hand end of a field).

Detailed description: The characters in the JCARD array are taken in pairs, starting with JCARD(J), and PACKed together into one element of KCARD, starting with KCARD(K). Since the characters are taken in pairs, an even number of characters will always be PACKed. If necessary, the character at JCARD(JLAST+1) will be used in order to make the last data PACKed a pair. More detailed information may be found in the PACK/UNPAC flowchart and listing.

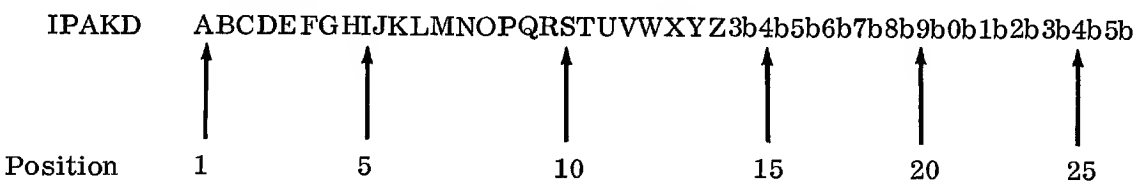
Example: DIMENSION IUNPK(26),IPAKD(26)

CALL PACK(IUNPK,1,25,IPAKD,1)

Before:

After:

IUNPK is the same.



Note that each two characters shown above represent one element of the array. Also, after IUNPK has been PACKed, the twenty-sixth character, Z, has been PACKed since 25 characters were specified (between J and JLAST).

Errors: None

Remarks: If JLAST is less than or equal to J, the first two characters of JCARD will be PACKed. An even number of characters in JCARD will always be PACKed into KCARD. An equation for how much space is required, in elements, in KCARD is

$$\text{Space in KCARD} = \left[\frac{\text{JLAST}-\text{J}+2}{2}\right]$$

This result is rounded down at all times.

PRINT

Format: CALL PRINT(JCARD,J,JLAST,NER)

Function: The printing of one line on the IBM 1132 Printer is initiated, and control is returned to the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the information to be printed, on the IBM 1132 Printer, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be printed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be printed (the right-hand end of a field).

NER - An integer variable. This variable indicates carriage tape channel conditions that have occurred in printing.

Detailed description: When the previous print operation is finished, if a print operation was going on, the routine begins. The characters to be printed are packed and reversed. Since the characters are taken in pairs, an even number of characters is required. If necessary, the character at JCARD(JLAST+1) will be used to get an even number. Then printing is initiated and control is returned to the user. When printing is finished, the printer spaces one line and the indicator, NER, is set as follows:

NER is	when
3	Channel 9 has been encountered
4	Channel 12 has been encountered

If channel 9 or channel 12 is not encountered, the indicator is not set.

If a WAIT occurs at location 41, one of the following conditions exists:

Condition	Accumulator (hex)
Printer not ready or end of forms.	6xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.	6xx1

All of the above WAITs require operator intervention.

Only one line can be printed at a time (JLAST-J+1 must be less than or equal to 120).

More detailed information may be found in the PRINT/SKIP flowchart and listing.

Example:

```
        DIMENSION IOUT(120)
        N=0
        CALL PRINT(IOUT,1,120,N)
        IF(N-3)  1,2,3
    2   Channel 9 routine
    3   Channel 12 routine
    1   Normal processing
```

The line in IOUT, from IOUT(1) through IOUT(120), is printed. The indicator is tested to see whether (1) the line was printed at channel 9 or (2) the line was printed at channel 12. Appropriate action will be taken.

Notice that the test of the indicator is made after printing. The test should always be performed in this way to see where the line has just been printed. If the indicator was set, the line was printed at channel 9 or channel 12.

Errors: If JLAST is less than J, only one character will be printed. If more than 120 characters are specified (JLAST-J+1 is greater than 120), only 120 characters will be printed.

Remarks: After each line is printed, the condition indicator should be checked for the channel 9 or channel 12 indication. In doing this the same variable should always be used for the indicator.

The indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

PUNCH

Format: CALL PUNCH(JCARD,J,JLAST,NER)

Function: Punches a card on the IBM 1442, Model 6 or 7. See Subroutine P1442 for punching on the 1442 Model 5.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the characters to be punched into a card, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be punched (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be punched (the right-hand end of a field).

NER - An integer variable. This variable indicates any conditions that have occurred in punching a card, and the nature of these conditions.

Detailed description: The characters to be punched are converted from EBCDIC to card codes, one at a time. When all characters have been converted, the punching operation is initiated. If an error occurs during the operation, the condition indicator is set, and the operation is continued. The possible values of the condition indicator and their meaning are listed below:

NER is	when
0	Last card condition.
1	Feed or punch check. Operator intervention required.

If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Punch not ready.	1xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your IBM representative. Save all output.	1xx1

All of the above WAITs require operator intervention.

Only one card can be punched at a time (JLAST-J+1 must be less than or equal to 80).

More detailed information may be found in the READ/PUNCH flowchart and listing.

Example:

```
DIMENSION IOTPT(80)

N=-1

CALL PUNCH(IOTPT,1,80,N)
```

Before:

```
IOTPT    NAME...ADDRESS...AMOUNT
           ↑         ↑         ↑
Position   1         20        60

N=-1
```

After:

IOTPT is the same.

N=0

The information in IOTPT, from IOTPT(1) to IOTPT(80), has been punched into a card. Since N=0, the information was punched correctly, and the card punched into was the last card.

Errors: If a punch or feed check occurs, the condition indicator will be set equal to 1. If an internal error occurs, the system will WAIT as specified above.

If more than 80 characters are specified (JLAST-J+1 is greater than 80), only 80 characters, one card, will be punched.

Remarks: After each card is punched, the condition indicator should be checked for the last card indication. This will occur only after the last card has physically been punched.

The condition indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

PUT

Format: CALL PUT(JCARD,J,JLAST,VAR,ADJST,N)

Function: Converts the whole portion of a real variable, VAR, to an EBCDIC integer number, half-adjusting as specified, and places the result, after decimal point alignment, in an array. An 11-zone is placed over the low-order, rightmost position in the array if VAR is negative.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array will contain the result of the PUT routine, EBCDIC coded information, in A1 format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the first position of JCARD to be filled with the result (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the last position to be filled with the result (the right-hand end of a field).

VAR - A real constant, a real expression, or a real variable. This is the number whose whole portion will be PUT.

ADJST - A real constant, a real expression, or a real variable. This is added to the variable, VAR, as a half-adjustment factor.

N - An integer constant, an integer expression, or an integer variable. This specifies the number of digits to truncate from the right-hand end of the number, VAR.

Detailed description: First, the half-adjustment factor is added to the real variable, VAR. Then, each digit is retrieved using the formula

EBCDIC DIGIT = 256 (BINARY DIGIT) - 4032

and placed in the output area. Each binary digit is retrieved by subtracting the digits already retrieved from VAR and multiplying by 10. The next digit is then retrieved and placed in the output area. More detailed information may be found in the PUT flowchart and listing.

Example: DIMENSION IPRNT(120)

CALL PUT(IPRNT, 1, 12, A, 5.0, 1)

Before:

A = 1234567.

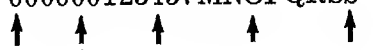
IPRNT ABCDEFGHIJKLMNOPQRSb

Position 1 5 10 15 20

After:

A = 1234567.

IPRNT 000000123457MNOPQRSb



Position 1 5 10 15 20

Errors: None

Remarks: If the receiving field, JCARD, is not large enough to hold all of the output, only the low-order digits are placed.

If JLAST is less than or equal to J, only one digit will be PUT.

It is necessary for the programmer to use the ADJST parameter in every PUT. For example, assume that the number to be PUT is 123.00. Because the IBM 1130 is a binary machine, the number may be represented in core storage as 122.999....If this number is PUT with ADJST equal to zero, the result will be 122. However, with ADJST equal to 0.5, the preliminary result is 123.499; when PUT, the result is 123. The value of ADJST should be a 5 in the decimal position one to the right of the low-order digit to be PUT.

The last two factors, ADJST and N, form a logical pair, and should usually appear as either:

	ADJST		N
	.5	and	0
or	5.	and	1
or	50.	and	2
or	500.	and	3
	etc.		etc.

ADJST should never be less than .5, since this will introduce fraction inaccuracies. From this it follows that N should never be negative.

If PUT (or GET) is used, the calling program must use extended precision.

P1403

Format: CALL P1403(JCARD,J,JLAST,NER)

Function: The printing of one line on the IBM 1403 Printer, Model 6 or 7, is initiated, and control is returned to the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the information to be printed, on the IBM 1403 Printer, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be printed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be printed (the right-hand end of a field).

NER - An integer variable. This variable indicates carriage control tape conditions that have occurred in printing.

Detailed description: When the previous print operation is finished, if a print operation was going on, the routine begins. The characters to be printed are converted to 1403 Printer codes and reversed so as to match the 1403 buffer mechanism. Since the characters are taken in pairs, an even number of characters is required. If necessary, the character at JCARD(JLAST+1) will be used to get an even number. Printing is then initiated and control is returned to the user. When printing is finished, the printer spaces one line and the indicator, NER, is set as follows:

NER is	when
3	Channel 9 has been encountered
4	Channel 12 has been encountered

If neither channel 9 nor channel 12 is encountered, the indicator is not set. If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Printer not ready or end of forms.	9000
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.	9001

All of the above WAITs require operator intervention.

Only one line can be printed at a time (JLAST-J+1 must be less than or equal to 120).

More detailed information may be found in the P1403 flowchart and listing.

```
Example:        DIMENSION IOUT(120)

                N=0

                CALL P1403(IOUT,1,120,N)

                IF(N-3)1,2,3

    2           Channel 9 routine

    3           Channel 12 routine

    1           Normal processing
```

The line in IOUT, from IOUT(1) through IOUT(120), is printed. The indicator is tested to see whether (1) the line was printed at channel 9 or (2) the line was printed at channel 12. Appropriate action will be taken.

Notice that the test of the indicator is made after printing. The test should always be performed in this way to see where the line has just been printed. If the indicator was set, the line was printed at channel 9 or channel 12.

Errors: If JLAST is less than J, two characters will be printed. If more than 120 characters are specified (JLAST-J+1 is greater than 120), only 120 characters will be printed.

Remarks: After each line is printed, the condition indicator should be checked for the channel 9 or channel 12 indication. In doing this, the same variable should always be used for the indicator.

The indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

This CSP subroutine uses three subprograms that are part of the Disk Monitor Version 2 subroutine library. If P1403 is to be used with Version 1 of the Monitor, ZIPCO, EBPT3, and PRNT3 must be loaded onto the Version 1 disk cartridge.

P1442

Format: CALL P1442(JCARD,J,JLAST,NER)

Function: Punches a card on the IBM 1442, Model 5, 6, or 7.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the characters to be punched into a card, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be punched (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be punched (the right-hand end of a field).

NER - An integer variable. This variable indicates any conditions that have occurred in punching a card, and the nature of these conditions.

Detailed description: The characters to be punched are converted from EBCDIC to card codes, one at a time. When all characters have been converted, the punching operation is initiated. If an error occurs during the operation, the condition indicator is set, and the operation is continued. The possible values of the condition indicator and their meaning are listed below:

NER is	when
0	Last card condition.
1	Feed or punch check. Operator intervention required.

If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Punch not ready.	1xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your IBM representative. Save all output.	1xx1

All of the above WAITs require operator intervention.

Only one card can be punched at a time (JLAST-J+1 must be less than or equal to 80).

More detailed information may be found in the P1442 flowchart and listing.

Example: DIMENSION IOTPT(80)

N = -1

CALL P1442(IOTPT,1,80,N)

Before:

IOTPT NAME...ADDRESS...AMOUNT

Position 1 20 60

N = -1

After:

IOTPT is the same.

N = 0

The information in IOTPT, from IOTPT(1) to IOTPT(80), has been punched into a card. Since N = 0, the information was punched correctly, and the card punched into was the last card.

Errors: If a punch or feed check occurs, the condition indicator will be set equal to 1. If an internal error occurs, the system will WAIT as specified above.

If JLAST is less than J, only one character will be punched.

If more than 80 characters are specified (JLAST-J+1 is greater than 80), only 80 characters, one card, will be punched.

Remarks: After each card is punched, the condition indicator may be checked for the last-card indication. This will occur only after the last card has physically been punched.

The condition indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

If a program contains no calls to the READ subroutine, this routine (P1442) may be used to punch cards on the 1442, Model 6 or 7, at a considerable savings in core storage. This is due to the fact that READ and PUNCH are two different entry points to the same subroutine. A call to one or both will cause the READ/PUNCH routine to be added to the core load. P1442 is smaller in size, since it is basically the PUNCH portion of the READ/PUNCH routine. A program may not CALL both READ/PUNCH and P1442; the Monitor will refuse to load two I/O routines that service the same device. To feed the first card, a P1442 CALL may be issued, punching 80 blanks.

This CSP subroutine uses part of the Disk Monitor Version 2 subroutine library. If P1442 is to be used with Version 1 of the Monitor, PNCH1 must be loaded onto the Version 1 disk cartridge.

Format: CALL READ(JCARD,J,JLAST,NER)

Function: Reads a card from the IBM 1442, Model 6 or 7, only, overlapping the conversion from card codes to EBCDIC.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. A card will be read into this array, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first word of JCARD into which a character will be read (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last word of JCARD into which a character will be read (the right-hand end of a field).

NER - An integer variable. This variable indicates any conditions that have occurred in reading a card, and the nature of these conditions.

Detailed description: A card read operation is started. While the card is being read, the characters, one at a time, are converted from card codes to EBCDIC. If an error occurs during the operation, the condition indicator is set, and the operation continues. The possible values of the condition indicator and their meaning are listed below:

NER is	when
0	Last card condition.
1	Feed or read check. Operator intervention required.

If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Reader not ready.	1xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your IBM representative. Save all output.	1xx1

All of the above WAITs require operator intervention.

Only one card can be read at a time (JLAST-J+1 must be less than or equal to 80). More detailed information may be found in the READ/PUNCH flowchart and listing.

Example:

```
DIMENSION INPUT(160)

N1=-1

CALL READ(INPUT,1,80,N1)

N2=-1

CALL READ(INPUT,81,160,N2)
```

Before:

```
INPUT    000000...0000000000
         ↑  ↑      ↑   ↑
Position 1  5      155 160
```

N1=-1

N2=-1

After:

```
INPUT    THIS IS THE NAME...SECOND CARD...
         ↑  ↑  ↑  ↑  ↑↑ ↑   ↑           ↑
Position 1  5  10 15 80 81 85  90        160
```

N1=-1

N2=-1

From the user's viewpoint the next card is read into the INPUT array (1-80). N1 is not one of the indicated values, so the first read was successful. The next card is read into the INPUT array (81-160). N2 is not one of the indicated values, so the second read was also successful.

Errors: If a read or feed check occurs, the condition indicator will be set equal to 1. If an internal error occurs, the system will WAIT as specified above.

If more than 80 characters are specified (JLAST-J+1 is greater than 80), only 80 characters, one card, will be read.

Remarks: After each card read, the condition indicator may be checked for the last card indication. This will occur only after the last card has physically been read into core storage.

The condition indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

Note that the READ subroutine will not detect Monitor // control cards, as opposed to the standard FORTRAN READ, which exits when such a card is encountered.

R2501

Format: CALL R2501(JCARD, J, JLAST, NER)

Function: Reads a card from the IBM 2501, Model A1 or A2 only, overlapping the conversion from card codes to EBCDIC.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. A card will be read into this array, in A1 format, one character per word. This array should always be 80 words in length.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first word of JCARD into which a character will be read (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last word of JCARD into which a character will be read (the right-hand end of a field).

NER - An integer variable. This variable indicates any conditions that have occurred in reading a card, and the nature of these conditions.

Detailed description: A card read operation is started. While the card is being read, the characters, one at a time, are converted from card codes to EBCDIC. If an error occurs during the operation, the condition indicator is set, and the operation continues. The possible values of the condition indicator and their meaning are listed below:

NER is	when
0	Last card condition.
1	Feed or read check. Operator intervention required.

If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Reader not ready.	1xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your IBM representative. Save all output.	1xx1

All of the above WAITs require operator intervention.

Only one card can be read at a time (JLAST-J+1 must be less than or equal to 80). More detailed information may be found in the R2501 flowchart and listing.

```
Example:    DIMENSION INPUT(160)

            N1=-1

            CALL R2501(INPUT,1,80,N1)

            N2=-1

            CALL R2501(INPUT,81,160,N2)
```

Before:

```
  INPUT     000000...0000000000
            ↑  ↑     ↑   ↑
Position    1  5     155 160

  N1=-1

  N2=-1
```

After:

```
  INPUT     THISbISbTHEbNAME...SECONDbCARD.....
            ↑  ↑  ↑   ↑  ↑  ↑  ↑  ↑     ↑
Position    1  5  10  15 80 81 85 90    160

  N1=-1

  N2=-1
```

The first card is read into the INPUT array (1-80). N1 is not one of the indicated values, so the first read was successful. The next card is read into the INPUT array (81-160). N2 is not one of the indicated values, so the second read was also successful.

Errors: If a read or feed check occurs, the condition indicator will be set equal to 1. If an internal error occurs, the system will WAIT as specified above.

If more than 80 characters are specified (JLAST-J+1 is greater than 80), only 80 characters, one card, will be read.

Remarks: After each card read, the condition indicator may be checked for the last-card indication. This will occur only after the last card has physically been read into core storage.

The condition indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

Note that the R2501 routine does not detect Monitor // control cards, as opposed to the standard FORTRAN READ, which exits when such a card is encountered.

This CSP subroutine uses part of the Disk Monitor Version 2 subroutine library. If R2501 is to be used with Version 1 of the Monitor, READ1 must be loaded onto the Version 1 disk cartridge.

SKIP

Format: CALL SKIP(N)

Function: Execute the requested control function on the IBM 1132 Printer

Parameter description:

N - An integer constant, an integer expression, or an integer variable. The value of this variable corresponds to an available control function.

Detailed description: If the printer is busy, the subroutine WAITs. Otherwise, or when the printer finishes, the routine executes the requested function and returns control to the calling program. The control functions and their values are as follows:

Function	Value
Immediate skip to channel 1	12544
Immediate skip to channel 2	12800
Immediate skip to channel 3	13056
Immediate skip to channel 4	13312
Immediate skip to channel 5	13568
Immediate skip to channel 6	13824
Immediate skip to channel 9	14592
Immediate skip to channel 12	15360
Immediate space of 1 space	15616
Immediate space of 2 spaces	15872
Immediate space of 3 spaces	16128
Suppress space after printing	0

Normal spacing is one space after printing.

Example: NUMBR=12544

CALL SKIP(NUMBR)

The carriage skips until a punch in channel 1 of the carriage control tape is encountered (normally this is at the top of a page).

Errors: Only the codes mentioned above can be used. The use of anything else will result in either no movement of the carriage or a WAIT at location 41 with 6xx1 in the accumulator (hex).

Remarks: When space suppression after printing is executed, it is reset to single-space after printing. If the user wishes to continue suppression, he must reissue the suppression command.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

Format: CALL STACK

Function: Selects the alternate stacker on the IBM 1442, Model 6 or 7, only for the next card to go through the punch station. More detailed information may be found in the STACK flowchart and listing.

Example: A card has been read. The sum of the four-digit numbers in columns 10-13 and 20-23 is punched in columns 1-5. If the sum is negative, the card should be selected into the alternate stacker. A program to solve the problem follows:

	FORTRAN Statement	Meaning
1	FORMAT(9X,I4,6X,I4)	Description of the input data.
2	FORMAT(I5)	Description of the output data.
	IO=2	Input unit number.
3	READ(IO,1)I1,I2	Input statement.
	I3=I1+I2	Sum.
	IF(I3)4,5,5	Is the sum negative?
4	CALL STACK	Yes — select the card.
5	WRITE(IO,2)I3	No — punch.
	GO TO 3	Process the next card.
	END	

Errors: None

Remarks: If the card reader is in a not-ready state (last card) and the card just read is to be stacker-selected, the card reader will not accept the stacker select command. The user should place a blank card after the card designating last card to his program. This will prevent the card reader from becoming not ready and will allow the card to be stacker-selected.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB ←
S1403
TYPER
UNPAC
WHOLE

SUB

Format: CALL SUB(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Subtracts one arbitrary-length decimal data field from another arbitrary-length decimal data field, placing the result in the second data field.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array that is subtracted, the subtrahend. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit to be subtracted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit to be subtracted (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array, the minuend, is subtracted from, and will contain the result in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of KCARD (the left-hand end of the field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of KCARD (the right-hand end of a field).

NER - An integer variable. Upon completion of the subroutine, this variable will indicate whether arithmetic overflow occurred.

Detailed description: The sign of the JCARD field is reversed and then the JCARD and KCARD fields are ADDed using the ADD subroutine. More detailed information may be found in the SUB flowchart and listing.

Example: DIMENSION IGRND(12), ITEM(6)

N=0

CALL SUB(ITEM,1,6,IGRND,1,12,N)

Before:

IGRND 000713665203 ITEM 102342

Position 1 5 10 Position 1 5

N=0

After:

IGRND 000713767545 ITEM is unchanged.

Position 1 5 10

N=0

The numeric data field ITEM, in decimal format, is SUBtracted from the numeric data field IGRND, also in decimal format. Note that the fields are both right-justified. In this case, since the ITEM field is negative, and the operation to be performed is subtraction, the ITEM field is added to the IGRND field. The error indicator, N, is the same, since there is no overflow out of the high-order digit, left-hand end, of the IGRND field.

Errors: If the KCARD field is not large enough to contain the sum (that is, if there is a carry out of the high-order digit), the error indicator, NER, will be set equal to KLAST.

If the JCARD field is longer than the KCARD field, nothing will be done and the error indicator will be equal to KLAST.

Remarks: See the remarks for the ADD subroutine.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403 ←
TYPER
UNPAC
WHOLE

S1403

Format: CALL S1403(N)

Function: Execute the requested control function on the IBM 1403 Printer, Model 6 or 7, only.

Parameter description:

N - An integer constant, an integer expression, or an integer variable. The value of this variable corresponds to an available control function.

Detailed description: If the printer is busy, the subroutine WAITs. Otherwise, or when the printer finishes, the routine executes the requested function and returns control to the calling program. The control functions and their values are as follows:

Function	Value
Immediate skip to channel 1	12544
Immediate skip to channel 2	12800
Immediate skip to channel 3	13056
Immediate skip to channel 4	13312
Immediate skip to channel 5	13568
Immediate skip to channel 6	13824
Immediate skip to channel 7	14080
Immediate skip to channel 8	14336
Immediate skip to channel 9	14592
Immediate skip to channel 10	14848
Immediate skip to channel 11	15104
Immediate skip to channel 12	15360
Immediate space of 1 space	15616
Immediate space of 2 spaces	15872
Immediate space of 3 spaces	16128
Suppress space after printing	0

Normal spacing is one space after printing.

Example: NUMBR=12544

CALL S1403(NUMBR)

The carriage skips until a punch in channel 1 of the carriage control tape is encountered. (Normally this is at the top of a page.)

Errors: Only the codes mentioned above can be used. The use of anything else will result in either no movement of the carriage or a WAIT at location 41 with 6xx1 in the accumulator (hex).

Remarks: When space suppression after printing is executed, it is reset to single-space after printing. If the user wishes to continue suppression, he must give the suppression command again.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

This CSP subroutine uses three subprograms that are part of the Disk Monitor Version 2 subroutine library. If S1403 is to be used with Version 1 of the Monitor, ZIPCO, EBPT3, and PRNT3 must be loaded onto the Version 1 disk cartridge.

TYPER

Format: CALL TYPER(JCARD,J,JLAST)

Function: The typing on the console printer is initiated, and control is returned to the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the characters to be printed on the console printer, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be printed (the left-hand end of a field).

JLAST - An integer constant, an integer variable, or an integer expression, greater than or equal to J. This is the position of the last character of JCARD to be printed (the right-hand end of a field).

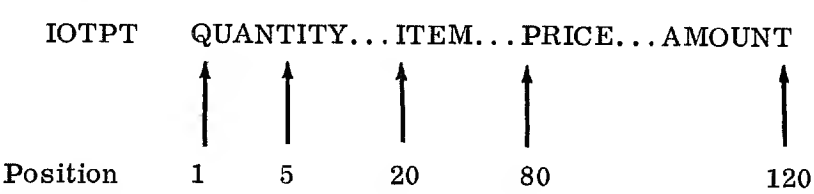
Detailed description: The characters to be printed are converted from EBCDIC to console printer codes and are packed. Since the characters are taken in pairs, an even number of characters is required. If necessary, the character at JCARD(JLAST+1) will be used to get an even number. Then the print operation is started. While printing is in progress, control is returned to the user's program.

More detailed information may be found in the TYPER/KEYBD flowchart and listing.

Example:

```
DIMENSION IOTPT(120)

CALL TYPER(IOTPT,1,120)
```

Before:



After:

IOTPT is the same. The line is being printed.

The printing of the line, specified in IOTPT, is initiated on the console printer, and control returns to the user's program.

Errors: If a WAIT occurs at location 41, one of the following conditions exists:

Condition	Accumulator (hex)
Console printer is not ready. Make it ready and continue.	2xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.	2xx1

If JLAST is less than J, two characters will be printed. If more than 120 characters are specified (JLAST-J+1 is greater than 120), only 120 characters will be printed.

Remarks: The asterisked characters in Appendix D of IBM 1130 Subroutine Library (C26-5925) are legal. No other characters will be printed.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O.

Control functions can be used on the console printer. The following table indicates the available control functions and the decimal constant required for each function:

Function	Decimal constant
Tabulate	1344
Shift to black	5184
Carrier return	5440
Backspace	5696
Line feed	9536
Shift to red	13632

The decimal constant corresponding to a particular function must be placed in the output area (JCARD). The function will take place when its position in the output area is printed.

Example:

```
JCARD(1)=5440

JCARD(21)=1344

JCARD(30)=5440

JCARD(51)=5440

JCARD(82)=5440

CALL TYPER(JCARD,1,101)
```

The above coding will carrier-return to a new line, then print characters 2-20 of JCARD, tab to the next tab stop; print characters 22-29, carrier return, print characters 31-50, carrier return, print characters 52-81, carrier return, and finally print characters 83-101.

Format: CALL UNPAC(JCARD,J,JLAST,KCARD,K)

Function: Information in A2 format, two characters per word, is UNPACked into A1 format, one character per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the input array, containing the data in A2 format, two characters per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first element of JCARD to be UNPACked (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable greater than or equal to J. This is the position of the last element of JCARD to be UNPACked (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is UNPACked, in A1 format, one character per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the UNPACked characters (the left-hand end of a field).

Detailed description: The characters in the JCARD array (A2) are UNPACked left to right, starting with JCARD(J), and placed in the KCARD array (A1), starting with KCARD(K). Each element of JCARD, when UNPACked, will require two elements of KCARD. More detailed information may be found in the PACK/UNPAC flowchart and listing.

Example: DIMENSION IUNPK(26),IPAKD(26)

CALL UNPAC(IPAKD,1,13,IUNPK,1)

Before:

IPAKD THISbINFORMATIONbWILLbUNPACKEDbbbbbbbbbbbbbbbbbbbbb

Position 1 5 10 15 20 25

IUNPK FbIbLbLbbbIbNbbbTbHbIbSbbbAbRbEbAbbbbbbbbbbbbbbbbbbb

Position 1 5 10 15 20 25

After:

IPAKD is the same.

IUNPK TbHbIbSbbbIbNbFbObRbMbAbTbIbObNbbbWbIbLbLbbbUbNbPbAb

Position 1 5 10 15 20 25

Note that each two characters shown above represent one element of the array.

Errors: None

Remarks: If JLAST is less than or equal to J, only the first element of JCARD,JCARD(J) will be UNPACked into the first two elements of KCARD. An even number of characters will always be UNPACked into KCARD. An equation for how much space is required, in elements, in KCARD is

$$\text{Space in KCARD} = 2\ (\text{JLAST}-\text{J}+1)$$

Format: WHOLE (EXPRS)

Function: Truncates the fractional portion of a real expression.

Parameter description:

EXPRS - A real expression. This is the expression that is truncated (the fractional part is made zero).

Detailed description: The result of the expression is shifted right until the fractional portion has been shifted off. Then the result is shifted left to give the original result with a zero fraction.

Example:	A=WHOLE(.1*B+.5)
Before:	
	A=0.0
	B=71234.99
After:	
	A=7123.000
	B=71234.99
The expression, (.1*B+.5), has been evaluated, and the fractional portion has been dropped.	

Errors: None

Remarks: The argument, EXPRS, must always be a real expression. If the purpose is to simply truncate the fraction from a number A, the expression must be (1.0*A).

If a single variable is used as an argument, the results of WHOLE are unpredictable. In other words, this will not work:

A=WHOLE(B)

Note that the WHOLE function truncates the value of the argument or expression within the parentheses; it does not round off before truncation. For this reason, the user must be careful when working with fractional numbers. For example, if

```
X = 1570000.
```

and

```
Y = WHOLE (X* .001)
```

Y will equal 1569.000 rather than 1570.000. This occurs because the multiplication by .001 yielded 1569.999 rather than 1570.000.

To avoid such a possibility, the argument for WHOLE should be half-adjusted by the user:

```
Y = WHOLE (X*.001+0.5)
```

before it is sent to WHOLE to be truncated.

SAMPLE PROBLEMS

PROBLEM 1

This program has been written to exercise many of the routines. A card is read and a code on that card initiates the operation of the specified routine. The card image is printed before execution of the routine, the resulting variable is printed and the card image is printed after execution of the routine.

Switch settings are as follows:

Input Device	Output Device	Switches		
		0	1	2
1442	console printer	down	down	down
1442	1132	up	down	down
1442	1403	up	up	down
2501	console printer	down	down	up
2501	1132	up	down	up
2501	1403	up	up	up

Make sure that the switches are set properly before the program begins.

After processing is completed, sample problem 1 will STOP with 1111 displayed in the accumulator. Press START to continue.

A general purpose *IOCS card

```
*IOCS(CARD,1132 PRINTER,TYPEWRITER)
```

has been supplied with the sample problem. If this does not match the 1130 configuration to be used, a new *IOCS card will be required.

Sample Problem 1: Source Program

```
// FOR                                                                  CSP2594D
** SAMPLE PROBLEM 1                                                     CSP25950
* NAME SMPL1                                                           CSP25960
*IOCS(CARD,1132 PRINTER,TYPEWRITER)                                    CSP25970
* ONE WORD INTEGERS                                                    CSP25980
* EXTENDED PRECISION                                                   CSP25990
* LIST ALL                                                             CSP26000
C-----GENERAL PURPOSE 1130 COMMERCIAL SUBROUTINE PACKAGE TEST PROGRAM. CSP26010
      DIMENSION NCARD(80), NAMES(5,13)                                  CSP26020
1     FORMAT (80A1)                                                     CSP26030
2     FORMAT (I1D, 4F10.0, F10.3)                                       CSP26040
3     FORMAT (30HONOW TESTING 1130 CSP ROUTINE ,5A1,16H WITH PARAMETERS,CSP26050
     X4F10.5, F10.3)                                                    CSP26060
4     FORMAT (13H CARD BEFORE=,80A1)                                    CSP26070
5     FORMAT (13H CARD AFTER =,80A1)                                    CSP26080
6     FORMAT(1H ,5I3,2X,12HCARD AFTER =,1X,80A1)                        CSP26090
7     FORMAT(1HD,4X,10HINDICATORS,3X,12HCARD BEFORE=,1X,80A1)           CSP26100
8     FORMAT (10H ANSWER IS, F20.3)                                     CSP26110
C-----DEFINE UNIT NUMBERS OF I/O DEVICES.                               CSP26120
      CALL DATSW(0,N)                                                   CSP26130
      CALL DATSW(1,M)                                                   CSP26140
      CALL DATSW(2,L)                                                   CSP26150
      NREAD=6*(1/L)+2                                                   CSP26160
      NWRIT=2*(1/N)+2*(1/M)+1                                           CSP26170
      READ (NREAD,1) NAMES                                              CSP26180
10    READ (NREAD,2) N, V1, V2, V3, V4, VAR                             CSP26190
      IF (N) 98,98,99                                                   CSP26200
98    STOP 1111                                                         CSP26210
99    WRITE (NWRIT,3) (NAMES(I,N), I=1,5), V1, V2, V3, V4, VAR          CSP26220
      N1=V1                                                             CSP26230
      N2=V2                                                             CSP26240
      N3=V3                                                             CSP26250
      N4=V4                                                             CSP26260
      NVAR=VAR                                                          CSP26270
      NER1=0                                                            CSP26280
      NER2=0                                                            CSP26290
      NER3=0                                                            CSP26300
      NER4=0                                                            CSP26310
      NER5=0                                                            CSP26320
      READ (NREAD,1) NCARD                                              CSP26330
      IF(N-7)  21,21,22                                                 CSP26340
21    WRITE(NWRIT,4) NCARD                                              CSP26350
C-----GO TO 1130 CSP ROUTINE                                            CSP26360
      GO TO (11,12,13,14,15,16,17), N                                   CSP26370
C-----COMP ROUTINE                                                      CSP26380
11    ANS=NCOMP(NCARD,N1,N2,NCARD,N3)                                   CSP26390
      GO TO 19                                                          CSP26400
C-----MOVE ROUTINE                                                      CSP26410
12    CALL MOVE(NCARD,N1,N2,NCARD,N3)                                   CSP26420
      GO TO 20                                                          CSP26430
C-----NZONE ROUTINE                                                     CSP26440
13    CALL NZONE(NCARD,N1,N2,N3)                                        CSP26450
      ANS=N3                                                            CSP26460
      GO TO 19                                                          CSP26470
C-----EDIT ROUTINE                                                      CSP26480
14    CALL EDIT(NCARD,N1,N2,NCARD,N3,N4)                                CSP26490
```

```
 SAMPLE PROBLEM 1                                                    PAGE 02

      GO TO 20                                                      CSP2650D
C-----GET ROUTINE                                                   CSP26510
15    ANS=GET(NCARD,N1,N2,V3)                                       CSP26520
      GO TO 19                                                      CSP26530
C-----PUT RDUTINE                                                   CSP26540
16    CALL PUT(NCARD,N1,N2,VAR,V3,N4)                               CSP26550
      GO TO 20                                                      CSP26560
C-----FILL RDUTINE                                                  CSP26570
17    CALL FILL(NCARD,N1,N2,NVAR)                                   CSP26580
      GO TO 20                                                      CSP26590
19    WRITE (NWRIT,8) ANS                                           CSP26600
20    WRITE (NWRIT,5) NCARD                                         CSP26610
      GO TO 10                                                      CSP26620
22    WRITE(NWRIT,7) NCARD                                          CSP26630
C-----A1DEC RDUTINE                                                 CSP26640
      CALL A1DEC(NCARD,N1,N2,NER1)                                  CSP26650
      CALL A1DEC(NCARD,N3,N4,NER2)                                  CSP26660
      N=N-7                                                         CSP26670
      GO TO (23,24,25,26,27,28),N                                   CSP26680
C-----ADD RDUTINE                                                   CSP26690
23    CALL ADD(NCARD,N1,N2,NCARD,N3,N4,NER3)                        CSP26700
      GO TO 29                                                      CSP26710
C-----SUB RDUTINE                                                   CSP26720
24    CALL SUB(NCARD,N1,N2,NCARD,N3,N4,NER3)                        CSP26730
      GO TO 29                                                      CSP26740
C-----MPY RDUTINE                                                   CSP26750
25    CALL MPY(NCARD,N1,N2,NCARD,N3,N4,NER3)                        CSP26760
      GO TO 29                                                      CSP26770
C-----DIV RDUTINE                                                   CSP26780
26    CALL DIV(NCARD,N1,N2,NCARD,N3,N4,NER3)                        CSP26790
      GO TO 29                                                      CSP26800
C-----ICDMP ROUTINE                                                 CSP26810
27    NER3=ICDMP(NCARD,N1,N2,NCARD,N3,N4)                           CSP26820
      GO TO 29                                                      CSP26830
C-----NSIGN RDUTINE                                                 CSP26840
28    CALL NSIGN(NCARD,N1,NVAR,NER3)                                CSP26850
C-----DECA1 RDUTINE                                                 CSP26860
29    CALL DECA1(NCARD,N1,N2,NER4)                                  CSP26870
      IF(N-3)  33,32,30                                             CSP26880
30    IF(N-4)  33,31,33                                             CSP26890
31    JSPAN=N2-N1                                                   CSP26900
      KSPAN=N4-N3                                                   CSP26910
      KSTRT=N3-JSPAN-1                                              CSP26920
      N3=N4-JSPAN                                                   CSP26930
      CALL DECA1(NCARD,KSTRT,N3-1,NER5)                             CSP26940
      GO TO 33                                                      CSP26950
32    N3=N3-N2+N1-1                                                 CSP26960
33    CALL DECA1(NCARD,N3,N4,NER5)                                  CSP26970
      WRITE(NWRIT,6) NER1,NER2,NER3,NER4,NER5,NCARD                 CSP26980
      GO TO 10                                                      CSP26990
      END                                                           CSP270D0
```

```
VARIABLE ALLOCATIONS
 V1    =0000  V2    =0D03  V3    =0006  V4    =0009  VAR   =000C  ANS   =000F  NCARD=0064  NAMES=00A5  N     =00A6  M     =00A7
 L     =00A8  NREAD=00A9  NWRIT=00AA  I     =00AB  N1    =00AC  N2    =00AD  N3    =00AE  N4    =00AF  NVAR  =00BD  NER1  =00B1
 NER2  =00B2  NER3  =DDB3  NER4  =00B4  NER5  =00B5  JSPAN=00B6  KSPAN=00B7  KSTRT=00B8

STATEMENT ALLOCATIONS
 1     =00C4  2     =0DC7  3     =00CC  4     =00EB  5     =00F6  6     =0101  7     =0111  8     =0126  10    =0177  98    =018A
 99    =018C  21    =D1E8  11    =01FA  12    =0206  13    =020F  14    =021C  15    =0226  16    =0230  17    =023A  19    =0242
 20    =0248  22    =0251  23    =0274  24    =027F  25    =028A  26    =0295  27    =02A0  28    =02AC  29    =02B2  30    =02C0
 31    =02C6  32    =02EE  33    =02F8

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION
 IOCS

CALLED SUBPROGRAMS
 DATSW   NCOMP   MOVE    NZONE   EOIT    GET     PUT     FILL    A1DEC   ADD     SUB     MPY     DIV     ICOMP   NSIGN
 DECA1   ELD     ESTD    IFIX    FLOAT   WRTYZ   SREO    SWRT    SCDMP   SFID    SIDAI   SIDIX   SIOF    SIOI    SUBSC
 STOP    CARDZ   PRNTZ

INTEGER CONSTANTS
     0=00BA       1=00BB       2=00BC       6=00BD    1111=00BE       5=00BF       7=00C0       3=00C1       4=00C2    4369=00C3

CORE REQUIREMENTS FOR SMPL1
 COMMON       0  VARIABLES    186  PROGRAM    600

END OF COMPILATION
```

Sample Problem 1: Output

```
// XEQ                                                    CSP27010

NOW TESTING 1130 CSP ROUTINE NCOMP WITH PARAMETERS   1.00000  10.00000  11.00000   0.00000     0.000
CARO BEFORE=ABCOEFGHIJKLMNOPQRST                                                    2CSP27040
ANSWER IS           -272.000
CARO AFTER =ABCOEFGHIJKLMNOPQRST                                                    2CSP27040

NOW TESTING 1130 CSP ROUTINE NCOMP WITH PARAMETERS   1.00000  10.00000  11.00000   0.00000     0.000
CARO BEFORE=BC80 F    BC80 F                                                        4CSP27060
ANSWER IS             0.000
CARO AFTER =BC80 F    BC80 F                                                        4CSP27060

NOW TESTING 1130 CSP ROUTINE NCOMP WITH PARAMETERS  20.00000  25.00000  30.00000   0.00000     0.000
CARO BEFORE=                  JKLMN     CBAFG                                     6CSP27080
ANSWER IS          224.000
CARO AFTER =                  JKLMN     CBAFG                                     6CSP27080

NOW TESTING 1130 CSP ROUTINE MOVE  WITH PARAMETERS   1.00000   5.00000  20.00000   0.00000     0.000
CARO BEFORE=ABCDE                                                                   8CSP27100
CARD AFTER =ABCDE             ABCDE                                                 8CSP27100

NOW TESTING 1130 CSP ROUTINE MOVE  WITH PARAMETERS  40.00000  49.00000   1.00000   0.00000     0.000
CARO BEFORE=                                9876543210                              10CSP27120
CARD AFTER =9876543210                      9876543210                              10CSP27120

NOW TESTING 1130 CSP ROUTINE N2ONE WITH PARAMETERS  10.00000   5.00000   0.00000   0.00000     0.000
CARO BEFORE=          A                                                             12CSP27140
ANSWER IS             1.000
CARO AFTER =          A                                                             12CSP27140

NOW TESTING 1130 CSP ROUTINE N2ONE WITH PARAMETERS  10.00000   5.00000   0.00000   0.00000     0.000
CARO BEFORE=          I                                                             14CSP27160
ANSWER IS             1.000
CARO AFTER =          I                                                             14CSP27160

NOW TESTING 1130 CSP ROUTINE N2ONE WITH PARAMETERS  20.00000   5.00000   0.00000   0.00000     0.000
CARO BEFORE=                  0                                                     16CSP27180
ANSWER IS             4.000
CARD AFTER =                  0                                                     16CSP27180

NOW TESTING 1130 CSP ROUTINE N2ONE WITH PARAMETERS  20.00000   5.00000   0.00000   0.00000     0.000
CARD BEFORE=                  9                                                     18CSP27200
ANSWER IS             4.000
CARD AFTER =                  9                                                     18CSP27200

NOW TESTING 1130 CSP ROUTINE N2ONE WITH PARAMETERS  30.00000   5.00000   0.00000   0.00000     0.000
CARD BEFORE=                              J                                         20CSP27220
ANSWER IS             2.000
CARD AFTER =                              J                                         20CSP27220

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   5.00000   0.00000   0.00000     0.000
CARD BEFORE=                              R                                         22CSP27240
ANSWER IS             2.000
CARD AFTER =                              R                                         22CSP27240

NOW TESTING 1130 CSP ROUTINE N2DNE WITH PARAMETERS  10.00000   1.00000   0.00000   0.00000     0.000
CARD BEFORE=          A                                                             24CSP27260
ANSWER IS             1.000
```

```
CARD AFTER =1234567              ********                                        48CSP27500

NOW TESTING 1130 CSP ROUTINE EDIT  WITH PARAMETERS   1.00000   6.00000  10.00000  30.00000     0.000
CARD BEFORE=00005M                    .*  .  CR                                 50CSP27520
CARD AFTER =000D5M    **************00.54CR                                     50CSP27520

NOW TESTING 1130 CSP ROUTINE EDIT  WITH PARAMETERS   1.00000   6.00000  20.00000  29.00000     0.000
CARD BEFORE=     5M                 ,0  .  -                                    52CSP27540
CARD AFTER =     5M                    .54-                                     52CSP27540

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   0.01000   0.00000     0.000
CARD BEFORE=12345                                                               54CSP27560
ANSWER IS              123.449
CARD AFTER =12345                                                               54CSP27560

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   0.01000   0.00000     0.000
CARD BEFORE=1234N                                                               56CSP27580
ANSWER IS             -123.449
CARD AFTER =1234N                                                               56CSP27580

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   7.00000   0.00100   0.00000     0.000
CARD BEFORE=1 3 5 7                                                             58CSP27600
ANSWER IS             1030.506
CARD AFTER =I 3 5 7                                                             58CSP27600

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   1.00000   0.00000     0.000
CARD BEFORE=12AB4                                                               60CSP27620
ANSWER IS                 0.000
CARD AFTER =12A84                                                               60CSP27620

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   1.00000   0.00000     0.000
CARD BEFORE=1230-                                                               62CSP27640
ANSWER IS          -12300.000
CARD AFTER =1230-                                                               62CSP27640

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   3.00000   0.00001   0.00000     0.000
CARD BEFORE=123                                                                 64CSP27660
ANSWER IS                 0.001
CARO AFTER =123                                                                 64CSP27660

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS   1.00000   5.00000   0.50000   0.00000 I2345.000
CARD BEFORE=                                                                    66CSP27680
CARD AFTER =12345                                                               66CSP27680

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS   1.00000   2.00000   5.00000   1.00000 12890.000
CARD BEFORE=                                                                    68CSP27700
CARD AFTER =B9                                                                  68CSP27700

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS  11.00000  15.00000   5.00000   1.00000 12345.000
CARO BEFORE=                                                                    70CSP27720
CARD AFTER =          01235                                                     70CSP27720

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS  10.00000  16.00000  50.00000   2.00000-34567.000
CARD BEFORE=                                                                    72CSP27740
CARD AFTER =          0000340                                                   72CSP27740

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS  10.00000  17.00000   5.00000   1.00000   -16.000
```

```
CARD AFTER =          A                                                         24CSP27260

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   1.00000   0.00000   0.00000     0.000
CARD BEFORE=          1                                                         26CSP27280
ANSWER IS                 4.000
CARD AFTER =          A                                                         26CSP27280

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   1.00000   0.00000   0.00000     0.000
CARD BEFORE=          J                                                         28CSP27300
ANSWER IS                 2.000
CARD AFTER =          A                                                         28CSP27300

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   4.00000   0.00000   0.00000     0.000
CARD BEFORE=                    I                                               30CSP27320
ANSWER IS                 1.000
CARD AFTER =                    9                                               30CSP27320

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   2.00000   0.00000   0.00000     0.000
CARD BEFORE=                    9                                               32CSP27340
ANSWER IS                 4.000
CARD AFTER =                    R                                               32CSP27340

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   3.00000   0.00000   0.00000     0.000
CARD BEFORE=                    R                                               34CSP27360
ANSWER IS                 2.000
CARD AFTER =                    Z                                               34CSP27360

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   3.00000   0.00000   0.00000     0.000
CARD BEFORE=                              D                                     36CSP27380
ANSWER IS                 1.000
CARD AFTER =                              U                                     36CSP27380

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   2.00000   0.00000   0.00000     0.000
CARD BEFORE=                              4                                     38CSP27400
ANSWER IS                 4.000
CARD AFTER =                              M                                     38CSP27400

NOW TESTING 113D CSP ROUTINE N2ONE WITH PARAMETERS  30.00000   4.00000   0.00000   0.00000     0.000
CARD BEFORE=                              M                                     40CSP27420
ANSWER IS                 2.D00
CARD AFTER =                              4                                     40CSP27420

NOW TESTING 1130 CSP ROUTINE EDIT  WITH PARAMETERS   1.00000   6.00000  20.00000  30.00000     0.000
CARD BEFORE=I23456                    ,  $.  CR                                 42CSP27440
CARD AFTER =123456                 $1.234.56                                    42CSP27440

NOW TESTING 1130 CSP ROUTINE EDIT  WITH PARAMETERS   1.00000   6.00000  20.00000  30.00000     0.000
CARD BEFORE=02343K                    ,  $.  CR                                 44CSP27460
CARD AFTER =02343K                  $234.32CR                                   44CSP27460

NOW TESTING 1130 CSP ROUTINE EDIT  WITH PARAMETERS   1.00000   6.00000  20.00000  29.00000     0.000
CARD BEFORE=00343-                    ,  $.  -                                  46CSP27480
CARD AFTER =00343-                     $34.30-                                  46CSP27480

NOW TESTING 1130 CSP ROUTINE EDIT  WITH PARAMETERS   1.00000   7.00000  21.00000  28.00000     0.000
CARD BEFORE=1234567                   ,  $.                                     48CSP27500
```

```
ARD BEFORE=                                                                    74CSP27760
CARD AFTER =          0000000K                                                 74CSP27760

NOW TESTING 1130 CSP ROUTINE FILL  WITH PARAMETERS   1.00000  10.00000   0.00000   0.00000 16448.000
CARD BEFORE=ABCDEFGHIJK                                                        76CSP27780
CARD AFTER =           K                                                       76CSP27780

NOW TESTING 1130 CSP ROUTINE FILL  WITH PARAMETERS  20.00000  25.00000   0.00000   0.00000 23360.000
CARD BEFORE=                  ABCDEFGH                                         78CSP27800
CARD AFTER =                  A$$$$$$H                                         78CSP27800

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              24                          2048  CSP27820
  0  0  0  0  0  CARD AFTER =                           00024                         02072  CSP27820

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              24                          2048  CSP27840
  0  0  0  0  0  CARD AFTER =                           00024                         02024  CSP27840

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              24                          2048  CSP27860
  0  0  0  0  0  CARD AFTER =                           00024                    0000048992  CSP27860

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              24                          2048  CSP27880
  0  0  0  0  0  CARD AFTER =                           00024                    0008500008  CSP27880

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              24                          2048  CSP27900
  0  0 -6  0  0  CARD AFTER =                           00024                         02048  CSP27900

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000      1.000

    INDICATORS   CARD BEFORE= 65                                                           CSP27920
  0  0  1  0  0  CARD AFTER = 65                                                           CSP27920

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              99                          2048  CSP27940
  0  0  0  0  0  CARD AFTER =                           00099                         02147  CSP27940

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              99                          2048  CSP27960
  0  0  0  0  0  CARD AFTER =                           00099                         01949  CSP27960

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              99                          2048  CSP27980
  0  0  0  0  0  CARD AFTER =                           00099                    0000202392  CSP27980

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              99                          2048  CSP28000
  0  0  0  0  D  CARD AFTER =                           00099                    0002000068  CSP28000

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

    INDICATORS   CARD BEFORE=                              99                          2048  CSP28020
  0  0 -9  0  0  CARD AFTER =                           00099                         02048  CSP28020

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000     -1.000

    INDICATORS   CARD BEFORE= 54                                                           CSP28040
  0  0  1  0  0  CARD AFTER = N4                                                           CSP28040

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                  12345678901234567890123456789012345678901234567890  CSP28060
  0  0  0  0  0  CARD AFTER = 1234567890123456789                   12345678902469135780246913578024691357 80  CSP28060

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                  12345678901234567890123456789012345678901234567890  CSP28080
  0  0  0  0  0  CARD AFTER = 1234567890123456789                   12345678900000000000000000000000000000000000000000  CSP28080

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                  12345678901234567890123456789012345678901234567890  CSP28100
  0  0  0  0  0  CARD AFTER = 1234567890123456789012345679081234567908123456791111111011111111110100  CSP28100

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                  12345678901234567890123456789012345678901234567890  CSP28120
  0  0  0  0  0  CARD AFTER = 1234567890123456789000000000000000000000010000000000000000000001234567890  CSP28120

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                  12345678901234567890123456789012345678901234567890  CSP28140
  0  0 -1  0  0  CARD AFTER = 1234567890123456789                   12345678901234567890123456789012345678901234567890  CSP28140

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000      0.000

    INDICATORS   CARD BEFORE= 32                                                           CSP28160
  0  0  1  0  0  CARD AFTER = L2                                                           CSP28160

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                  12345678901234567890123456789012345678901234567890  CSP28180
  0  0  0  0  0  CARD AFTER = 1234567890123456789-                  12345678900000000000000000000000000000000000000000  CSP28180

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                  12345678901234567890123456789012345678901234567890  CSP28200
  0  0  0  0  0  CARD AFTER = 1234567890123456789-                  12345678902469135780246913578024691357 80  CSP28200

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                  12345678901234567890123456789012345678901234567890  CSP28220
  0  0  0  0  0  CARD AFTER = 1234567890123456789-012345679081234567908123456791111111011111111110-  CSP28220

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 123456789012345678-                   12345678901234567890123456789012345678901234567890  CSP28240
  0  0  0  0  0  CARD AFTER = 12345678901234567890000000000000000000010000000000-0000000000123456789 0  CSP28240

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000
```

```
   INDICATORS   CARD BEFORE= 1234567890123456789-                    12345678901234567890123456789-  CSP28260
 0  0 -1  D  0  CARD AFTER = 1234567890123456789-                    12345678901234567890123456789-  CSP28260

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000   1.000

   INDICATORS   CARD BEFORE= ON                                                                      CSP28280
 0  0 -1  0  0  CARD AFTER = 6N                                                                      CSP28280

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                   123456789012345678901234567890-  CSP28300
 0  0  0  0  0  CARD AFTER = 12345678901234567890                   123456789000000000000000000000-  CSP28300

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                   123456789012345678901234567890-  CSP28320
 0  0  0  0  0  CARD AFTER = 12345678901234567890                   123456789024691357802469135780-  CSP28320

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                   123456789012345678901234567890-  CSP28340
 0  0  0  0  0  CARD AFTER = 1234567890123456789001234567908123456790812345679111111101111111110-  CSP28340

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                   123456789012345678901234567890-  CSP28360
 0  0  0  0  0  CARD AFTER = 1234567890123456789000000000000000000000100000000-000000000123456789-  CSP28360

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                   123456789012345678901234567890-  CSP28380
 0  0  0  0  0  CARD AFTER = 12345678901234567890                   123456789012345678901234567890-  CSP28380

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000  -1.000

   INDICATORS   CARD BEFORE= NM                                                                      CSP28400
 0  0 -1  0  0  CARD AFTER = NM                                                                      CSP28400

NOW TESTING 1130 CSP ROUTINE A00   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                   123456789012345678901234567890-  CSP28420
 0  0  0  0  0  CARD AFTER = 1234567890123456789-                   123456789024691357802469135780-  CSP28420

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                   123456789012345678901234567890-  CSP28440
 0  0  0  0  0  CARD AFTER = 1234567890123456789-                   123456789000000000000000000000-  CSP28440

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                   123456789012345678901234567890-  CSP28460
 0  0  0  0  0  CARD AFTER = 1234567890123456789-012345679081234567908123456791111111011111111110100  CSP28460

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                   123456789012345678901234567890-  CSP28480

 0  0  0  0  0  CARD AFTER = 1234567890123456789-000000000000000000100000000000000000000123456789-  CSP28480

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                   123456789012345678901234567890-  CSP28500
 0  0  0  0  0  CARD AFTER = 1234567890123456789-                   123456789012345678901234567890-  CSP28500

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000   0.000

   INDICATORS   CARD BEFORE= ML                                                                      CSP28520
 0  0 -1  0  0  CARD AFTER = 4L                                                                      CSP28520

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                                 1234567890123456789  CSP28540
 0  0  0  0  0  CARD AFTER = 12345678901234567890                                 2469135780246913578  CSP28540

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                                 1234567890123456789  CSP28560
 0  0  0  0  0  CARD AFTER = 12345678901234567890                                 0000000000000000000  CSP28560

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                                 1234567890123456789  CSP28580
 0  0  0  0  0  CARD AFTER = 12345678901234567890          012345679081234567911111110111111111110100  CSP28580

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                                 1234567890123456789  CSP28600
 0  0  0  0  0  CARD AFTER = 12345678901234567890          000000000000000000010000000000000000000  CSP28600

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 12345678901234567890                                 1234567890123456789  CSP28620
 0  0  0  0  0  CARD AFTER = 12345678901234567890                                 1234567890123456789  CSP28620

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000   1.000

   INDICATORS   CARD BEFORE= -0                                                                      CSP28640
 0  0 -1  0  0  CARD AFTER = 00                                                                      CSP28640

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                                 1234567890123456789  CSP28660
 0  0  0  0  0  CARD AFTER = 1234567890123456789-                                 0000000000000000000  CSP28660

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                                 1234567890123456789  CSP28680
 0  0  0  0  0  CARD AFTER = 1234567890123456789-                                 2469135780246913578  CSP28680

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000   0.000

   INDICATORS   CARD BEFORE= 1234567890123456789-                                 1234567890123456789  CSP28700
 0  0  0  0  0  CARD AFTER = 1234567890123456789-          012345679081234567911111110111111111010-  CSP28700
```

```
 NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0  0  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28720
                                 00000000000000000000J00000000000000000000-  CSP28720

 NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0***  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28740
                                                                   1234567890123456789-  CSP28740

 NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS  1.00000   1.00000   2.00000   2.00000    -1.000

    INDICATORS   CARD BEFORE= -0                                                                   
  0  0 -1  0  0  CARD AFTER = -0                                                                   CSP28760
                                                                                                  CSP28760

 NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                                 
  0  0  0  0  0  CARD AFTER = 12345678901234567890                1234567890123456789-  CSP28780
                                                                   0000000000000000000-  CSP28780

 NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                                 
  0  0  0  0  0  CARD AFTER = 12345678901234567890                1234567890123456789-  CSP28800
                                                                   2469135780246913578-  CSP28800

 NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                                 
  0  0  0  0  0  CARD AFTER = 12345678901234567890                1234567890123456789-  CSP28820
                                 01234567908123456791111111011111111110-  CSP28820

 NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                                 
  0  0  0  0  0  CARD AFTER = 12345678901234567890                1234567890123456789-  CSP28840
                                 00000000000000000000J00000000000000000000-  CSP28840

 NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                                 
  0  0***  0  0  CARD AFTER = 12345678901234567890                1234567890123456789-  CSP28860
                                                                   1234567890123456789-  CSP28860

 NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS  1.00000   1.00000   2.00000   2.00000      0.000

    INDICATORS   CARD BEFORE= -0                                                                   
  0  0 -1  0  0  CARD AFTER = 00                                                                   CSP28880
                                                                                                  CSP28880

 NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0  0  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28900
                                                                   2469135780246913578-  CSP28900

 NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0  0  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28920
                                                                   0000000000000000000-  CSP28920

 NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0  0  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28940
                                 01234567908123456791111111011111111110100  CSP28940

 NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0  0  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28960
                                 00000000000000000001000000000000000000-  CSP28960

 NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000      0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                                                
  0  0  0  0  0  CARD AFTER = 12345678901234567890-                1234567890123456789-  CSP28980
                                                                   1234567890123456789-  CSP28980
```

Sample Problem 1: Data Input Listing

```
// XEQ                                                                  CSP27010
NCOMPMOVE N2ONEE0IT GET  PUT  FILL A0D  SUB  MPY  0IV  ICOMPNSIGN       CSP27020
        1         1         10         11                              1CSP27030
A8C0EFGHIJKLMNOPORST                                                    2CSP27040
        1         1         10         11                              3CSP27050
8C80 F   8C80 F                                                         4CSP27060
        1        20         25         30                              5CSP27070
                  JKLMN      C8AFG                                      6CSP27080
        2         1          5         20                              7CSP27090
A8C0E                                                                   8CSP27100
        2        40         49          1                              9CSP27110
                                        9876543210                     10CSP27120
        3        10          5                                         11CSP27130
        A                                                              12CSP27140
        3        10          5                                         13CSP27150
        I                                                              14CSP27160
        3        20          5                                         15CSP27170
                  0                                                    16CSP27180
        3        20          5                                         17CSP27190
                  9                                                    18CSP27200
        3        30          5                                         19CSP27210
                             J                                         20CSP27220
        3        30          5                                         21CSP27230
                             R                                         22CSP27240
        3        10          1                                         23CSP27250
        A                                                              24CSP27260
        3        10          1                                         25CSP27270
        1                                                              26CSP27280
        3        10          1                                         27CSP27290
        J                                                              28CSP27300
        3        20          4                                         29CSP27310
                  I                                                    30CSP27320
        3        20          2                                         31CSP27330
                  9                                                    32CSP27340
        3        20          3                                         33CSP27350
                  R                                                    34CSP27360
        3        30          3                                         35CSP27370
                             0                                         36CSP27380
        3        30          2                                         37CSP27390
                             4                                         38CSP27400
        3        30          4                                         39CSP27410
                             M                                         40CSP27420
        4         1          6         20         30                   41CSP27430
123456              ,  $.  CR                                          42CSP27440
        4         1          6         20         30                   43CSP27450
02343K              ,  $.  CR                                          44CSP27460
        4         1          6         20         29                   45CSP27470
00343-              ,  $.  -                                           46CSP27480
        4         1          7         21         28                   47CSP27490
1234567             ,  $.                                              48CSP27500
        4         1          6         10         30                   49CSP27510
00005M              ,*  .  CR                                          50CSP27520
        4         1          6         20         29                   51CSP27530
    5M              ,0  .  -                                           52CSP27540
        5         1          5        .01                             53CSP27550
12345                                                                  54CSP27560
        5         1          5        .01                             55CSP27570
1234N                                                                  56CSP27580
        5         1          7       .001                             57CSP27590
1 3 5 7                                                                58CSP27600
```

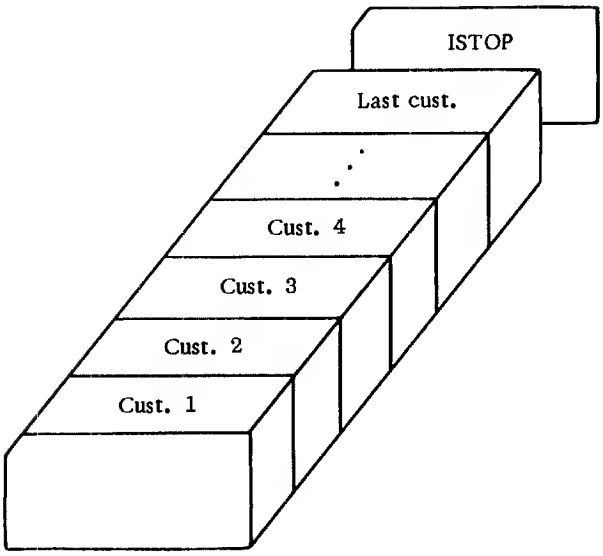
```
         5         1         5        1.                                        59CSP27610
12AB4
         5         1         5        1.                                        60CSP27620
                                                                                61CSP27630
1230-
         5         1         3   .00001                                        62CSP27640
                                                                                63CSP27650
123
         6         1         5       0.5         0    12345.                  64CSP27660
                                                                                65CSP27670
         6         1         2       5.0         1    12890.                  66CSP27680
                                                                                67CSP27690
         6        11        15       5.0         1    12345.                  68CSP27700
                                                                                69CSP27710
         6        10        16      50.0         2   -34567.                  70CSP27720
                                                                                71CSP27730
         6        10        17       5.0         1      -16.                  72CSP27740
                                                                                73CSP27750
         7         1        10                             16448.              74CSP27760
ABCDEFGHIJK                                                                     75CSP27770
         7        20        25                             23360.              76CSP27780
                  ABCDEFGH                                                      77CSP27790
                                                                                78CSP27800
        08        31        35        66        70                              CSP27810
                                      24                                2048    CSP27820
        09        31        35        66        70                              CSP27830
                                      24                                2048    CSP27840
        10        31        35        66        70                              CSP27850
                                      24                                2048    CSP27860
        11        31        35        66        70                              CSP27870
                                      24                                2048    CSP27880
        12        31        35        66        70                              CSP27890
                                      24                                2048    CSP27900
        13         1         1         2         2         1.                   CSP27910
65                                                                              CSP27920
        08        31        35        66        70                              CSP27930
                                      99                                2048    CSP27940
        09        31        35        66        70                              CSP27950
                                      99                                2048    CSP27960
        10        31        35        66        70                              CSP27970
                                      99                                2048    CSP27980
        11        31        35        66        70                              CSP27990
                                      99                                2048    CSP28000
        12        31        35        66        70                              CSP28010
                                      99                                2048    CSP28020
        13         1         1         2         2        -1.                   CSP28030
54                                                                              CSP28040
        08        01        20        41        70                              CSP28050
12345678901234567890                    12345678901234567890                    CSP28060
        09        01        20        41        70                              CSP28070
12345678901234567890                    12345678901234567890                    CSP28080
        10        01        20        41        70                              CSP28090
12345678901234567890                    12345678901234567890                    CSP28100
        11        01        20        41        70                              CSP28110
12345678901234567890                    12345678901234567890                    CSP28120
        12        01        20        41        70                              CSP28130
12345678901234567890                    12345678901234567890                    CSP28140
        13         1         1         2         2                              CSP28150
32                                                                              CSP28160
        08        01        20        41        70                              CSP28170
```

```
12345678901234567890-                                                  CSP28180
        09          01          20          41          70             CSP28190
12345678901234567890-                                                  CSP28200
        10          01          20          41          70             CSP28210
12345678901234567890-                                                  CSP28220
        11          01          20          41          70             CSP28230
12345678901234567890-                                                  CSP28240
        12          01          20          41          70             CSP28250
12345678901234567890-                                                  CSP28260
        13           1           1           2           2         1.   CSP28270
ON                                                                     CSP28280
        08          01          20          41          70             CSP28290
12345678901234567890                                                   CSP28300
        09          01          20          41          70             CSP28310
12345678901234567890                                                   CSP28320
        10          01          20          41          70             CSP28330
12345678901234567890                                                   CSP28340
        11          01          20          41          70             CSP28350
12345678901234567890                                                   CSP28360
        12          01          20          41          70             CSP28370
12345678901234567890                                                   CSP28380
        13           1           1           2           2        -1.   CSP28390
NM                                                                     CSP28400
        08          01          20          41          70             CSP28410
12345678901234567890-                                                  CSP28420
        09          01          20          41          70             CSP28430
12345678901234567890-                                                  CSP28440
        10          01          20          41          70             CSP28450
12345678901234567890-                                                  CSP28460
        11          01          20          41          70             CSP28470
12345678901234567890-                                                  CSP28480
        12          01          20          41          70             CSP28490
12345678901234567890-                                                  CSP28500
        13           1           1           2           2             CSP28510
ML                                                                     CSP28520
        08          01          20          51          70             CSP28530
12345678901234567890                                                   CSP28540
        09          01          20          51          70             CSP28550
12345678901234567890                                                   CSP28560
        10          01          20          51          70             CSP28570
12345678901234567890                                                   CSP28580
        11          01          20          51          70             CSP28590
12345678901234567890                                                   CSP28600
        12          01          20          51          70             CSP28610
12345678901234567890                                                   CSP28620
        13           1           1           2           2         1.   CSP28630
-0                                                                     CSP28640
        08          01          20          51          70             CSP28650
12345678901234567890-                                                  CSP28660
        09          01          20          51          70             CSP28670
12345678901234567890-                                                  CSP28680
        10          01          20          51          70             CSP28690
12345678901234567890-                                                  CSP28700
        11          01          20          51          70             CSP28710
12345678901234567890-                                                  CSP28720
        12          01          20          51          70             CSP28730
12345678901234567890-                                                  CSP28740
        13           1           1           2           2        -1.   CSP28750
-0                                                                     CSP28760
        08          01          20          51          70             CSP28770
12345678901234567890                                                   CSP28780
```

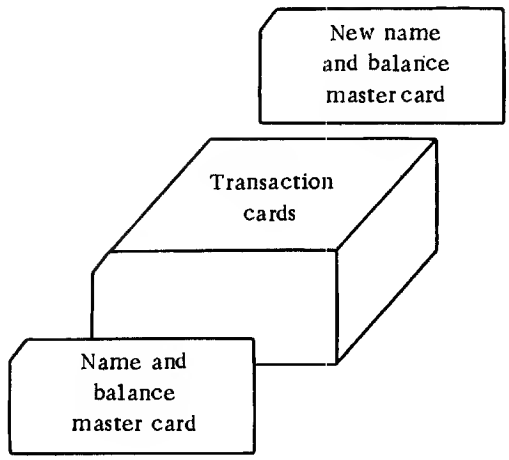
```
        09          01          20          51          70             CSP28790
12345678901234567890                                                   CSP28800
        10          01          20          51          70             CSP28810
12345678901234567890                                                   CSP28820
        11          01          20          51          70             CSP28830
12345678901234567890                                                   CSP28840
        12          01          20          51          70             CSP28850
12345678901234567890                                                   CSP28860
        13           1           1           2           2             CSP28870
-0                                                                     CSP28880
        08          01          20          51          70             CSP28890
12345678901234567890-                                                  CSP28900
        09          01          20          51          70             CSP28910
12345678901234567890-                                                  CSP28920
        10          01          20          51          70             CSP28930
12345678901234567890-                                                  CSP28940
        11          01          20          51          70             CSP28950
12345678901234567890-                                                  CSP28960
        12          01          20          51          70             CSP28970
12345678901234567890-                                                  CSP28980
                                                                       CSP28990
```

PROBLEM 2

The purpose of this program is to create invoices. The input deck is as follows:



Input deck



Detailed description of individual customer deck

Each customer has the old master name and balance card, followed by the transaction cards, followed by a blank master name and balance card. The invoice is printed as in the example, and a new master name and balance card image is printed on the console printer. Then the next customer is processed until the stop code card is reached (ISTOP in cc 1-5). In an actual situation the new card image would be punched and stacker-selected. Then, as input to the next run of the program, a new input deck would have to be prepared.

Switch settings are the same as for sample problem 1, except that output cannot be directed toward the console printer.

Input Device	Output Device	Switches		
		0	1	2
1442	1132	up	down	down
1442	1403	up	up	down
2501	1132	up	down	up
2501	1403	up	up	up

Make sure that the switches are set properly before the program begins.

After processing is completed, sample problem 2 will STOP with 0111 displayed in the accumulator. Press START to continue.

Note: Sample Problem 2 cannot be executed if Version 1 of the Monitor is being used.

Sample Problem 2: Detailed Description

1. Read all constant information and determine output unit (1132 or 1403).

2. Initialize error indicators.

   a. J=2

   b. I=0, L=0, M=0

3. Read the first card. It should be a master card.

4. Is the card read in 3 the last card?

   No — 5 Yes — 64

5. Is the card read in 3 above a master card?

   No — 72 Yes — 6

6. Go to the top of a new page.

7. Clear the print area.

8. Print the customer name.

9. Move the edit mark to the work area.

10. Edit the previous balance.

11. Print the customer street address.

12. Move the words PREVIOUS BALANCE to the print area.

13. Move the work area to the print area.

14. Print the customer city, state, and zip code.

15. Skip 3 lines.

16. Print the column headings.

17. Print the print area.

18. Clear the print area.

19. Convert the previous balance from A1 format to decimal format.

20. Is the conversion in 19 correct?

    No — 66 Yes — 21

21. Set the total (ISUM) equal to the previous balance.

22. Set up the output area for the new master card.

23. Read a card.

24. Is the card read at 23 the last card?

    No — 25 Yes — 64

25. Is the card read at 23 a master card?

    No — 26 Yes — 52

26. Is the card read at 23 a transaction card?

    No — 49 Yes — 27

27. Is the card read at 23 for the same customer being processed?

    No — 49 Yes — 28

28. Move the item name to the print area.

29. Move the edit mask to the print area for dollar amount.

30. Move the edit mask to the print area for quantity.

31. Edit the quantity.

32. Edit the dollar amount.

33. Print the detail line assembled in 28 through 32.

34. Has channel 12 on the carriage tape been encountered?

    No — 35 Yes — 46

35. Convert the dollar amount from A1 format to decimal format.

36. Is the conversion in 35 correct?

    No — 40 Yes — 37

37. Add the dollar amount to ISUM.

38. Did overflow occur in the addition in 37?

    No — 23 Yes — 39

39. STOP and display 777.

40. Make the character in error a digit.

41. Try to convert only the character in error.

42. Is the conversion in 41 correct?

    No — 43 Yes — 44

43. STOP and display 666.

44. Convert the entire field back to A1 format.

45. Go to 35.

46. Go to the top of a new page.

47. Print the headings.

48. Go to 35.

49. Type ERROR on the console printer.

50. Type the card read on the console printer.

51. Go to 23.

52. Convert the total (ISUM) from decimal format to A1 format.

53. Is the conversion in 52 correct?

    No — 54 Yes — 55

54. STOP and display 555.

55. Clear the print area.

56. Move the edit mask to the print area.

57. Edit the total (ISUM).

58. Place the unedited total (ISUM) in the new master card.

59. Type the new master card image on the console printer.

60. Move the word TOTAL to the print area.

61. Skip 2 lines.

62. Print the print area, the total line.

63. Go to 2b.

64. Type END OF JOB.

65. STOP and display 111.

66. Make the character in error a digit.

67. Try to convert only the character in error.

68. Is the conversion in 67 correct?

    No — 69 Yes — 70

69. STOP and display 444.

70. Convert the entire field back to A1 format.

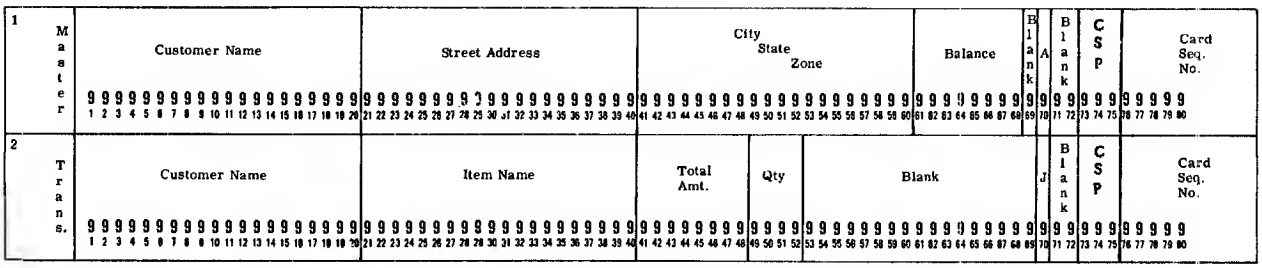
71. Go to 19.

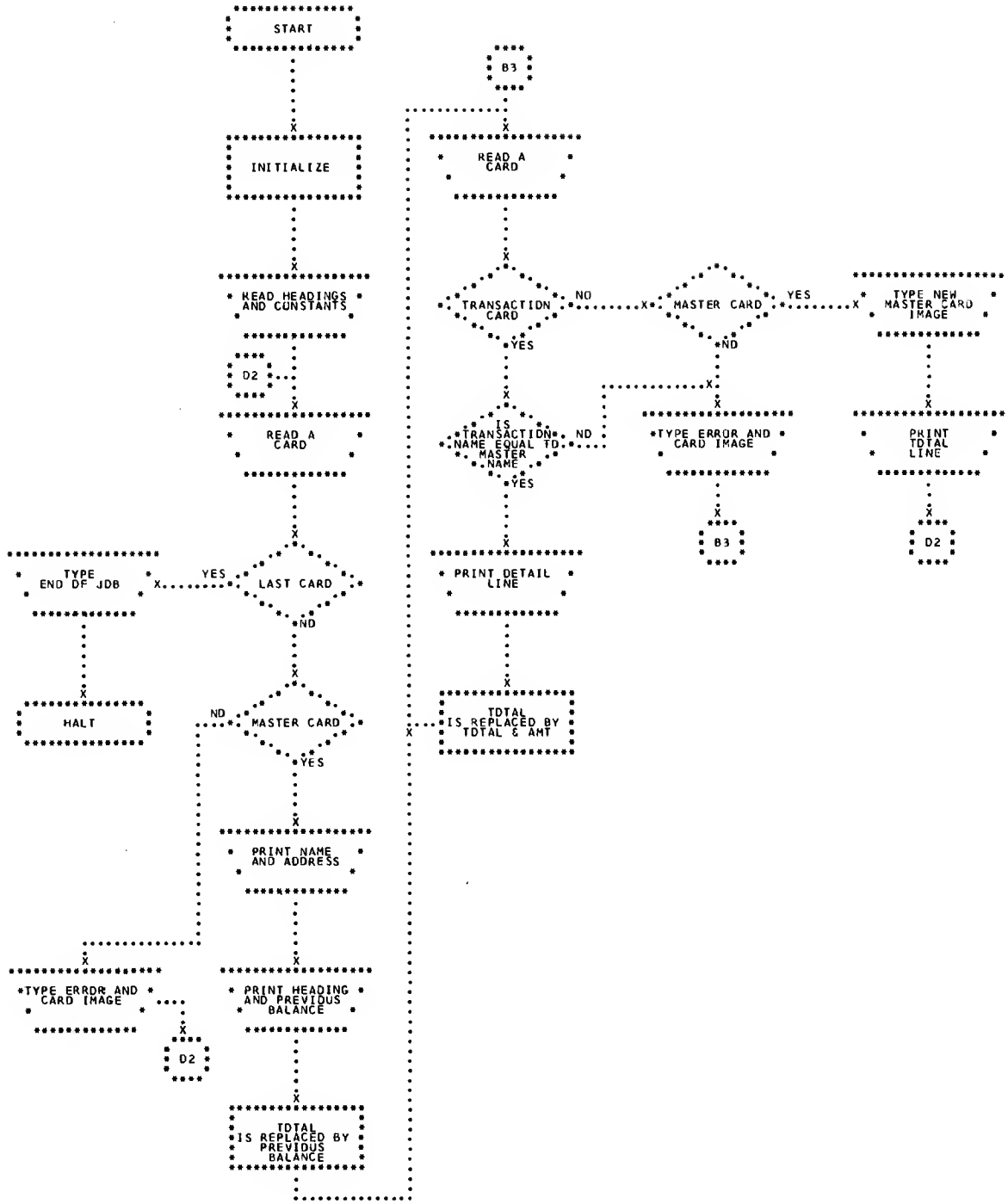
72. Type ERROR on the console printer.

73. Type the card read on the console printer.

74. Go to 2b.

Card Formats




START
INITIALIZE
READ HEADINGS AND CONSTANTS
D2
READ A CARD
TYPE END OF JOB
YES
LAST CARD
ND
HALT
ND
MASTER CARD
YES
PRINT NAME AND ADDRESS
TYPE ERROR AND CARD IMAGE
D2
PRINT HEADING AND PREVIOUS BALANCE
TOTAL IS REPLACED BY PREVIOUS BALANCE
B3
READ A CARD
TRANSACTION CARD
NO
MASTER CARD
YES
TYPE NEW MASTER CARD IMAGE
YES
ND
IS TRANSACTION NAME EQUAL TO MASTER NAME
ND
TYPE ERROR AND CARD IMAGE
PRINT TOTAL LINE
YES
B3
D2
PRINT DETAIL LINE
TOTAL IS REPLACED BY TOTAL & AMT

Sample Problem 2: Source Program

```
// FOR                                                                  CSP29000
** SAMPLE PROBLEM 2                                                     CSP29010
* NAME SMPL2                                                           CSP29020
* LIST ALL                                                             CSP29030
* ONE WORD INTEGERS                                                    CSP29040
* EXTENDED PRECISION                                                   CSP29050
C-----THE INPUT IS MADE UP OF A MASTER CARD FOLLOWED BY THE TRANSACTION CSP29060
C-----CARDS FOR EACH CUSTOMER. WE WANT TO PRINT AN INVOICE AND PRINT A  CSP29070
C-----NEW MASTER CARD FOR EACH CUSTOMER.                                CSP29080
      DIMENSION INCRD(82),IMASK(13),IPRNT(79),IOTCO(80),ISTOP(5),       CSP29090
     1IHEAD(80), IPRVB(16),ITOT(5),IWK(13),ISUM(8),IEROR(6),IEOJ(10)    CSP29100
      CALL DATSW (2,N2)                                                 CSP29110
      CALL DATSW(1,N3)                                                  CSP29120
      GO TO (28,27),N2                                                  CSP29130
27    CALL READ(IEOJ,1,10,J)                                            CSP29140
      CALL READ(IEROR,1,6,J)                                            CSP29150
      CALL READ(IMASK,1,13,J)                                           CSP29160
      CALL READ(IPRVB,1,16,J)                                           CSP29170
      CALL READ(IHEAD,1,72,J)                                           CSP29180
      CALL READ(IHEAD,73,80,J)                                          CSP29190
      CALL READ(ISTOP,1,5,J)                                            CSP29200
      CALL READ(ITOT,1,5,J)                                             CSP29210
      GO TO 58                                                          CSP29220
28    CALL R2501(IEOJ,1,10,J)                                           CSP29230
      CALL R2501(IEROR,1,6,J)                                           CSP29240
      CALL R2501(IMASK,1,13,J)                                          CSP29250
      CALL R2501(IPRVB,1,16,J)                                          CSP29260
      CALL R2501(IHEAD,1,72,J)                                          CSP29270
      CALL R2501(IHEAD,73,80,J)                                         CSP29280
      CALL R2501(ISTOP,1,5,J)                                           CSP29290
      CALL R2501(ITOT,1,5,J)                                            CSP29300
58    J=2                                                               CSP29310
      INCRD(81)=16448                                                   CSP29320
      INCRD(82)=5440                                                    CSP29330
1     I=0                                                               CSP29340
      L=0                                                               CSP29350
      M=0                                                               CSP29360
      GO TO (30,29),N2                                                  CSP29370
29    CALL READ(INCRD,1,80,J)                                           CSP29380
      GO TO 59                                                          CSP29390
30    CALL R2501(INCRD,1,80,J)                                          CSP29400
59    IF(J-1)  22,2,2                                                   CSP29410
2     IF(NCOMP(INCRD,1,5,ISTOP,1))   3,22,3                             CSP29420
3     CALL NZONE(INCRD,70,5,K)                                          CSP29430
      IF(K-1)   26,4,26                                                 CSP29440
4     GO TO (34,33) ,N3                                                 CSP29450
33    CALL SKIP(12544)                                                  CSP29460
      GO TO 60                                                          CSP29470
34    CALL S14D3(12544)                                                 CSP29480
60    CALL FILL(IPRNT,1,79,16448)                                       CSP29490
      GO TO (36,35),N3                                                  CSP29500
35    CALL PRINT(INCRD,1,20,I)                                          CSP29510
      GO TO 61                                                          CSP29520
36    CALL P1403(INCRD,1,20,I)                                          CSP29530
61    CALL MOVE(IMASK,1,13,IWK,1)                                       CSP29540
      CALL EDIT(INCRD,61,68,IWK,1,13)                                   CSP29550
```

```
      GO TO (38,37),N3                                                  CSP29560
37    CALL PRINT(INCRO,21,40,I)                                         CSP29570
      GO TO 62                                                          CSP29580
38    CALL PI403(INCRO,21,40,I)                                         CSP29590
62    CALL MOVE(IPRVB,1,16,IPRNT,23)                                    CSP29600
      CALL MOVE(IWK,1,13,IPRNT,67)                                      CSP29610
      GO TO (41,39),N3                                                  CSP29620
39    CALL PRINT(INCRO,41,60,I)                                         CSP29630
      CALL SKIP(16128)                                                  CSP29640
      CALL PRINT(IHEAO,1,80,I)                                          CSP29650
      CALL PRINT(IPRNT,1,79,I)                                          CSP29660
      GO TO 63                                                          CSP29670
41    CALL P1403(INCRO,41,60,I)                                         CSP29680
      CALL S1403(16128)                                                 CSP29690
      CALL P1403(IHEAO,1,80,I)                                          CSP29700
      CALL P1403(IPRNT,1,79,I)                                          CSP29710
63    CALL FILL(IPRNT,I,79,16448)                                       CSP29720
40    CALL A10EC(INCRO,61,68,L)                                         CSP29730
      IF(L)   5,5,23                                                    CSP29740
5     CALL MOVE(INCRO,61,68,ISUM,1)                                     CSP29750
      CALL MOVE(INCRD,I,80,IOTCO,1)                                     CSP29760
6     GO TO (32,31),N2                                                  CSP29770
31    CALL REAO(INCRO,1,80,J)                                           CSP29780
      GO TO 64                                                          CSP29790
32    CALL R2501(INCRD,1,80,J)                                          CSP29800
64    IF(J-1)   22,7,7                                                  CSP29810
7     CALL N2ONE(INCRO,70,5,K)                                          CSP29820
      IF(K-1)   18,19,8                                                 CSP29830
8     IF(K-2)   18,9,18                                                 CSP29840
9     IF(NCOMP(INCRO,1,20,IOTCO,1))   18,10,18                          CSP29850
10    CALL MOVE(INCRO,21,40,IPRNT,23)                                   CSP29860
      CALL MOVE(IMASK,1,13,IPRNT,67)                                    CSP29870
      CALL MOVE(IMASK,3,8,IPRNT,7)                                      CSP29880
      IPRNT(12)=-4032                                                   CSP29890
      CALL EDIT(INCRO,49,52,IPRNT,7,12)                                 CSP29900
      CALL EDIT(INCRO,41,48,IPRNT,67,79)                                CSP29910
      GO TO(49,48),N3                                                   CSP29920
48    CALL PRINT(IPRNT,1,79,I)                                          CSP29930
      GO TO 65                                                          CSP29940
49    CALL PI403(IPRNT,1,79,I)                                          CSP29950
65    IF(I-3)   11,11,17                                                CSP29960
11    CALL A10EC(INCRO,41,48,L)                                         CSP29970
      IF(L)   12,12,14                                                  CSP29980
12    CALL A00(INCRO,41,48,ISUM,18,M)                                   CSP29990
      IF(M)   13,6,13                                                   CSP30000
13    CALL IONO                                                         CSP30010
      STOP 777                                                          CSP30020
14    CALL N2ONE(INCRO,L,4,N1)                                          CSP30030
      N1=0                                                              CSP30040
      CALL A10EC(INCRD,L,L,NI)                                          CSP30050
      IF(N1)   16,16,15                                                 CSP30060
15    CALL IONO                                                         CSP30070
      STOP 666                                                          CSP30080
16    CALL OECA1(INCRO,41,48,L)                                         CSP30090
```



```
      L=0                                                               CSP30100
      GO TO 11                                                          CSP30110
17    GO TO (51,50),N3                                                  CSP30120
50    CALL SKIP(12544)                                                  CSP30130
      CALL PRINT(IHEAO,1,80,I)                                          CSP30140
      GO TO 66                                                          CSP30150
51    CALL S1403(12544)                                                 CSP30160
      CALL P1403(IHEAO,1,80,I)                                          CSP30170
66    I=0                                                               CSP30180
      GO TO 11                                                          CSP30190
18    CALL TYPER(IEROR,1,5)                                             CSP30200
      CALL TYPER(INCRO,1,82)                                            CSP30210
      GO TO 6                                                           CSP30220
19    CALL OECA1(ISUM,I,8,L)                                            CSP30230
      IF(L)   20,21,20                                                  CSP30240
20    CALL IONO                                                         CSP30250
      STOP 555                                                          CSP30260
21    CALL FILL(IPRNT,I,79,16448)                                       CSP30270
      CALL MOVE(IMASK,1,13,IPRNT,67)                                    CSP30280
      CALL EDIT(ISUM,1,8,IPRNT,67,79)                                   CSP30290
      CALL MOVE(ISUM,1,8,IOTCO,61)                                      CSP30300
      CALL TYPER(IOTCO,1,80)                                            CSP30310
      CALL MOVE(ITOT,1,5,IPRNT,23)                                      CSP30320
      GO TO (55,54),N3                                                  CSP30330
54    CALL SKIP(15872)                                                  CSP30340
      CALL PRINT(IPRNT,1,79,I)                                          CSP30350
      GO TO 67                                                          CSP30360
55    CALL S1403(15872)                                                 CSP30370
      CALL P1403(IPRNT,1,79,I)                                          CSP30380
67    CALL TYPER(INCRO,81,82)                                           CSP30390
      GO TO 1                                                           CSP30400
22    CALL TYPER(IEOJ,1,10)                                             CSP30410
      CALL IONO                                                         CSP30420
      STOP 111                                                          CSP30430
23    CALL N2ONE(INCRO,L,4,N1)                                          CSP30440
      N1=0                                                              CSP30450
      CALL A10EC(INCRO,L,L,N1)                                          CSP30460
      IF(N1)   25,25,24                                                 CSP30470
24    CALL IONO                                                         CSP30480
      STOP 444                                                          CSP30490
25    CALL OECA1(INCRO,61,68,L)                                         CSP30500
      L=0                                                               CSP30510
      GO TO 40                                                          CSP30520
26    CALL TYPER(IEROR,1,5)                                             CSP30530
      CALL TYPER(INCRO,1,82)                                            CSP30540
      GO TO 1                                                           CSP30550
      END                                                               CSP30560
```

```
VARIABLE ALLOCATIONS
 INCRO=0051  IMASK=005E  IPRNT=00A0  IOTCD=00F0  ISTOP=0102  IHEAO=0152  IPRVB=0162  ITOT =0167  IWK  =0174  ISUM =017C
 IEROR=0182  IEOJ =018C  N2   =0180  N3   =018E  J    =018F  I    =0190  L    =019I  M    =0192  K    =0193  NI   =0194

STATEMENT ALLOCATIONS
  27   =0106  28   =0208  58   =0238  1    =0248  29   =025A  30   =0262  59   =0268  2    =026E  3    =0277  4    =0283
```

```
 SAMPLE PROBLEM 2                                                    PAGE 04
 33   =0289  34   =028E  60   =0291  35   =029D  36   =02A5  61   =02AB  37   =02C0  38   =02C8  62   =02CE  39   =02E2
 41   =02F9  63   =030E  40   =0314  5    =031E  6    =032C  31   =0332  32   =033A  64   =0340  7    =0346  8    =0354
 9    =035A  10   =0363  48   =0395  49   =039D  65   =03A3  11   =03A9  12   =03B3  13   =03C0  14   =03C4  15   =03D8
 16   =030C  17   =03E8  50   =03EE  51   =03F9  66   =0402  18   =040B  19   =0414  20   =041E  21   =0422  54   =0450
 55   =045B  67   =0464  22   =046B  23   =0474  24   =0488  25   =04BC  26   =0498

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 DATSW   READ    R2501   NCOMP   NZONE   SKIP    S1403   FILL    PRINT   P1403   MOVE    EDIT    A1DEC   ADD     IDNO
 DECA1   TYPER   STOP

INTEGER CONSTANTS
     2=0198       1=0199      10=019A       6=019B      13=019C      16=019D      72=019E      73=019F      80=01A0       5=01A1
 16448=01A2    5440=01A3       0=01A4      70=01A5   12544=01A6      79=01A7      20=01A8      61=01A9      68=01AA      21=01AB
    40=01AC      23=01A0      67=01AE      41=01AF      60=01B0   16128=01B1       3=01B2       8=01B3       7=01B4    4032=01B5
    49=01B6      52=01B7      12=01B8      48=01B9     777=01BA       4=01BB     666=01BC      82=01BD     555=01BE   15872=01BF
    81=01C0     111=01C1     444=01C2    1911=01C3    1638=01C4    1365=01C5     273=01C6    1092=01C7

CORE REQUIREMENTS FOR SMPL2
 COMMON       0  VARIABLES     408  PROGRAM     780

END OF COMPILATION
```

```
// XEQ                                                    CSP30570
```

Sample Problem 2: Invoice Output

DAVES MARKET
1997 WASHINGTON ST.
NEWTOWN, MASS. 02158

QTY	NAME	AMT
	PREVIOUS BALANCE	$111.29
8	SUGAR - BAGS	$21.02
11	CHICKEN SOUP - CASES	$38.76
10	TOMATO SOUP - CASES	$30.11
8	SUGAR RETURNED	$21.02CR
6	COOKIES - CASES	$45.21
17	GINGER ALE - CASES	$52.37
17	ROOT BEER - CASES	$52.37
17	ORANGE AOE - CASES	$52.37
17	CREME SODA - CASES	$52.37
17	CHERRY SODA - CASES	$52.37
17	SODA WATER - CASES	$52.37
25	DOG FOOD - CASES	$101.26
25	CAT FOOD - CASES	$101.26
10	SOAP POWDER - CASES	$72.89
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORNED BEEF	$33.75
12	ROAST BEEF	$33.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
50	MILK - GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORNED BEEF	$33.75
12	ROAST BEEF	$33.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
50	MILK - GALS	$57.42
100	MILK - HALF GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
1,000	BREAD - LOAF	$150.00

QTY	NAME	AMT
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
50	MILK - GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORNED BEEF	$33.75
12	ROAST BEEF	$33.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
100	MILK - HALF GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
	TOTAL	$3,893.25

STANDISH MOTORS
10 WATER STREET
PLYMOUTH, MASS.02296

QTY	NAME	AMT
	PREVIOUS BALANCE	$2,356.36
20	AIR CLEANERS - CASES	$200.03
6	GREASE - BARRELS	$165.24
20	TIRES - 650 X 13	$260.38
50	TIRES - 750 X 14	$900.53
50	TIRES - 800 X 14	$1,012.00
100	GASOLINE CAPS	$99.68
	TOTAL	$4,994.22

Sample Problem 2: Console Printer Log and New Master Card Listing

```
ERROR THIS IS A OELIBERATE ERROR                                          J  CSP30660
ERROR DAVE MARKET                THIS CARO IS A OELIBERATE MISTAKE        J  CSP30680
OAVES MARKET        1997 WASHINGTON ST. NEWTOWN, MASS. 0215800389325 A  CSP30670
ERROR STANOISH MOTOR       THIS CARD IS NOT CORRECT   ABCOEFGHIJKLMNOPQRSTUVJ  CSP31470
STANOISH MOTORS     10 WATER STREET     PLYMOUTH, MASS.0229600499422 A  CSP31410

ENO OF JOB
```

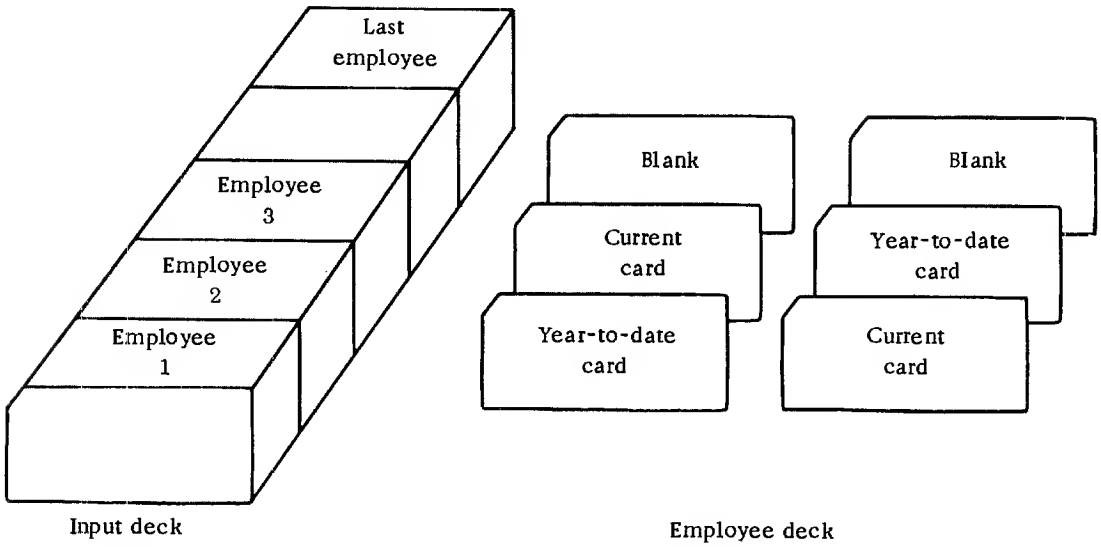
Sample Problem 2: Data Input Listing

```
// XEQ                                                                    CSP30570
END OF JOB                                                                CSP30580
ERROR                                                                     CSP30590
    ,  $.  CR                                                             CSP30600
PREVIOUS BALANCE                                                          CSP30610
       QTY          NAME                                                  CSP30620
AMT                                                                       CSP30630
ISTOP                                                                     CSP30640
TOTAL                                                                     CSP30650
THIS IS A DELIBERATE ERROR                                             J  CSP30660
OAVES MARKET        1997 WASHINGTON ST. NEWTOWN, MASS. 02158000111129  A  CSP30670
DAVE MARKET               THIS CARD IS A DELIBERATE MISTAKE            J  CSP30680
OAVES MARKET        SUGAR - BAGS        000021020008                   J  CSP30690
DAVES MARKET        CHICKEN SOUP - CASES000038760011                   J  CSP30700
OAVES MARKET        TOMATO SOUP - CASES 000030110010                   J  CSP30710
OAVES MARKET        SUGAR RETURNED      0000210K0008                   J  CSP30720
OAVES MARKET        COOKIES - CASES     000045210006                   J  CSP30730
OAVES MARKET        GINGER ALE - CASES  000052370017                   J  CSP30740
OAVES MARKET        ROOT BEER - CASES   000052370017                   J  CSP30750
OAVES MARKET        DRANGE ADE - CASES  000052370017                   J  CSP30760
OAVES MARKET        CREME SODA - CASES  000052370017                   J  CSP30770
OAVES MARKET        CHERRY SODA - CASES 000052370017                   J  CSP30780
OAVES MARKET        SODA WATER - CASES  000052370017                   J  CSP30790
OAVES MARKET        DOG FOOD - CASES    000101260025                   J  CSP30800
OAVES MARKET        CAT FOOD - CASES    000101260025                   J  CSP30810
OAVES MARKET        SOAP POWDER - CASES 000072890010                   J  CSP30820
OAVES MARKET        DETERGENT - CASES   000072890010                   J  CSP30830
DAVES MARKET        HAM - TINS          000036750012                   J  CSP30840
OAVES MARKET        HAM - LOAF          000033750012                   J  CSP30850
OAVES MARKET        SALAMI              000033750012                   J  CSP30860
OAVES MARKET        BOLOGNA             000033750012                   J  CSP30870
OAVES MARKET        CORNED BEEF         000033750012                   J  CSP30880
DAVES MARKET        ROAST BEEF          000033750012                   J  CSP30890
OAVES MARKET        BREAD - LOAF        000150001000                   J  CSP30900
OAVES MARKET        ROLLS               000150004000                   J  CSP30910
DAVES MARKET        MILK - QUARTS       000057420200                   J  CSP30920
DAVES MARKET        MILK - HALF GALS    000057420100                   J  CSP30930
OAVES MARKET        MILK - GALS         000057420050                   J  CSP30940
OAVES MARKET        POTATOES - BAGS     000011230100                   J  CSP30950
DAVES MARKET        TOMATOES - LOOSE    000011230100                   J  CSP30960
OAVES MARKET        CARROTS - BUNCHES   000011230100                   J  CSP30970
OAVES MARKET        DETERGENT - CASES   000072890010                   J  CSP30980
OAVES MARKET        HAM - TINS          000036750012                   J  CSP30990
OAVES MARKET        HAM - LOAF          000033750012                   J  CSP31000
OAVES MARKET        SALAMI              000033750012                   J  CSP31010
OAVES MARKET        BOLOGNA             000033750012                   J  CSP31020
DAVES MARKET        CORNED BEEF         000033750012                   J  CSP31030
OAVES MARKET        ROAST BEEF          000033750012                   J  CSP31040
DAVES MARKET        BREAD - LOAF        000150001000                   J  CSP31050
OAVES MARKET        ROLLS               000150004000                   J  CSP31060
OAVES MARKET        MILK - QUARTS       000057420200                   J  CSP31070
DAVES MARKET        MILK - GALS         000057420050                   J  CSP31080
OAVES MARKET        MILK - HALF GALS    000057420100                   J  CSP31090
OAVES MARKET        POTATOES - BAGS     000011230100                   J  CSP31100
OAVES MARKET        TOMATOES - LOOSE    000011230100                   J  CSP31110
OAVES MARKET        CARROTS - BUNCHES   000011230100                   J  CSP31120
OAVES MARKET        DETERGENT - CASES   000072890010                   J  CSP31130
OAVES MARKET        HAM - TINS          000036750012                   J  CSP31140
OAVES MARKET        BREAD - LOAF        000150001000                   J  CSP31150
OAVES MARKET        ROLLS               000150004000                   J  CSP31160
```

```
OAVES MARKET        MILK - QUARTS       000057420200                   J  CSP31170
DAVES MARKET        MILK - HALF GALS    000057420100                   J  CSP31180
OAVES MARKET        MILK - GALS         000057420050                   J  CSP31190
OAVES MARKET        POTATOES - BAGS     000011230100                   J  CSP31200
OAVES MARKET        TOMATOES - LOOSE    000011230100                   J  CSP31210
OAVES MARKET        CARROTS - BUNCHES   000011230100                   J  CSP31220
OAVES MARKET        DETERGENT - CASES   000072890010                   J  CSP31230
OAVES MARKET        HAM - TINS          000036750012                   J  CSP31240
OAVES MARKET        HAM - LOAF          000033750012                   J  CSP31250
OAVES MARKET        SALAMI              000033750012                   J  CSP31260
OAVES MARKET        BOLOGNA             000033750012                   J  CSP31270
DAVES MARKET        CORNED BEEF         000033750012                   J  CSP31280
OAVES MARKET        ROAST BEEF          000033750012                   J  CSP31290
OAVES MARKET        BREAD - LOAF        000150001000                   J  CSP31300
OAVES MARKET        ROLLS               000150004000                   J  CSP31310
OAVES MARKET        MILK - QUARTS       000057420200                   J  CSP31320
OAVES MARKET        MILK - HALF GALS    000057420100                   J  CSP31330
OAVES MARKET        MILK - HALF GALS    000057420100                   J  CSP31340
OAVES MARKET        POTATOES - BAGS     000011230100                   J  CSP31350
OAVES MARKET        TOMATOES - LOOSE    000011230100                   J  CSP31360
DAVES MARKET        CARROTS - BUNCHES   000011230100                   J  CSP31370
OAVES MARKET        DETERGENT - CASES   000072890010                   J  CSP31380
OAVES MARKET        HAM - TINS          000036750012                   J  CSP31390
                                                                       A  CSP31400
STANDISH MOTORS     10 WATER STREET     PLYMOUTH, MASS.0229600235636   A  CSP31410
STANDISH MOTORS     AIR CLEANERS - CASES000200030020                   J  CSP31420
STANDISH MOTORS     GREASE  -  BARRELS  000165240006                   J  CSP31430
STANDISH MOTORS     TIRES  -  650 X 13  000260380020                   J  CSP31440
STANDISH MOTORS     TIRES  -  750 X 14  000900530050                   J  CSP31450
STANDISH MOTORS     TIRES  -  800 X 14  001012000050                   J  CSP31460
STANDISH MOTOR      THIS CARD IS NOT CORRECT   ABCDEFGHIJKLMNOPQRSTUVJ  CSP31470
STANDISH MOTORS     GASOLINE CAPS       000099680100                   J  CSP31480
                                                                       A  CSP31490
ISTOP                                                                     CSP31500
```

PROBLEM 3

The purpose of this program is to print a payroll register and punch a new year-to-date card for each employee. The input deck is as follows:



The year-to-date and current cards are read and processed. The payroll register is printed as in the example, and a new year-to-date card image is printed on the console printer. Then the next employee is processed.

As is shown, the order of the year-to-date card and current card is not known before the cards are read.

Switch settings are as follows:

Input Device	Output Device	Switches		
		0	1	2
1442	console printer	down	down	down
1442	1132	up	down	down
1442	1403	up	up	down
2501	console printer	down	down	up
2501	1132	up	down	up
2501	1403	up	up	up

Make sure that the switches are set properly before the program begins.

After processing is completed, sample problem 3 will STOP with 3333 displayed in the accumulator. Press START to continue.

A general purpose *IOCS card has been supplied with the sample problem. If this does not match the 1130 configuration to be used, a new *IOCS card will be required.

*IOCS (CARD, 1132 PRINTER, TYPEWRITER)

Sample Problem 3: Detailed Description

1. Determine the output unit from the data switches.

   Console printer, 1132 Printer, or 1403 Printer

2. Read the edit mask.

3. Read a card.

4. Is the card read in (3) blank?

   Yes — 18 No — 5

5. Is the card read in (3) a year-to-date card?

   Yes — 11 No — 6

6. Is the card read in (3) a current card?

   Yes — 8 No — 7

7. Stop.

8. Move the employee number to storage (JEMP).

9. Extract the number of hours worked (HRS).

10. Go to (3).

11. Move the department number to storage (IDEP).

12. Move the employee number to storage (IEMP).

13. Move the employee name to storage (INM).

14. Move the Social Security number to storage (ISS).

15. Move the pay rate to storage (IRT).

16. Move the year-to-date gross to storage (IYTD).

17. Go to (3).

18. Are IEMP and JEMP the same?

    Yes — 19 No — 24

19. Current amount (CURR) is set equal to HRS times pay rate.

20. New year-to-date is set equal to CURR +IYTD.

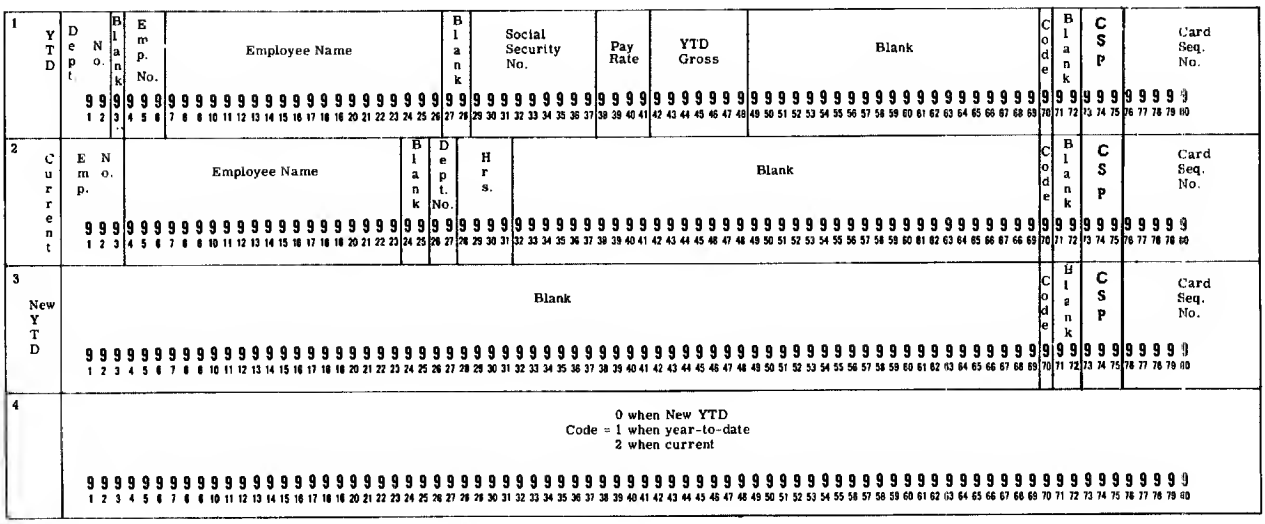
21. Print a new year-to-date card image on the console printer.

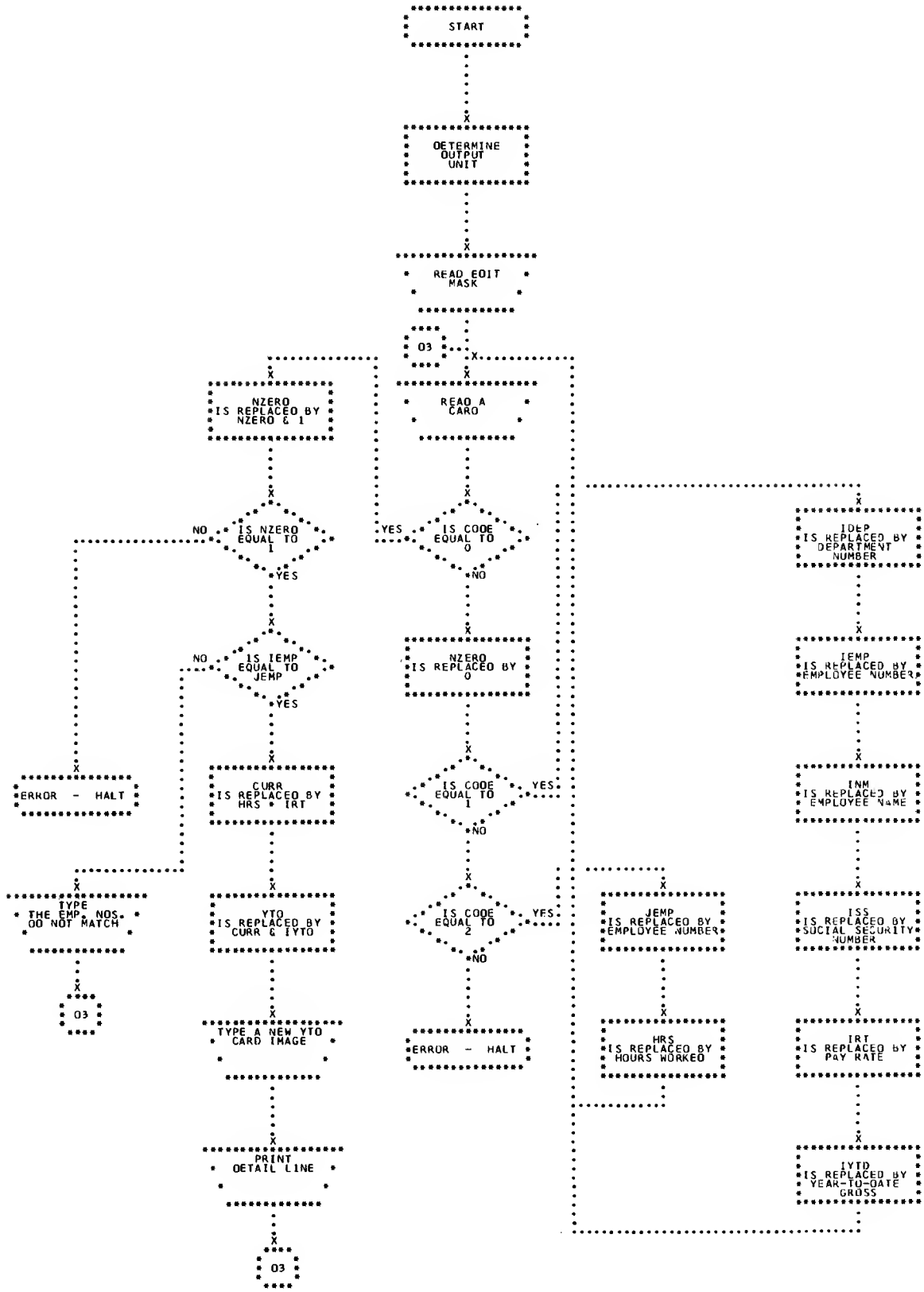
22. Print the payroll register line as in the example.

23. Go to (3).

24. Halt. If start is pushed, go to (3).

Card Formats




START
DETERMINE OUTPUT UNIT
READ EDIT MASK
03
NZERO IS REPLACED BY NZERO & 1
READ A CARD
NO
IS NZERO EQUAL TO 1
YES
YES
IS CODE EQUAL TO 0
NO
NO
IS IEMP EQUAL TO JEMP
YES
NZERO IS REPLACED BY 0
ERROR - HALT
CURR IS REPLACED BY HRS * IRT
IS CODE EQUAL TO 1
YES
NO
TYPE THE EMP. NOS. DO NOT MATCH
YTO IS REPLACED BY CURR & IYTO
IS CODE EQUAL TO 2
YES
NO
03
TYPE A NEW YTO CARD IMAGE
ERROR - HALT
PRINT DETAIL LINE
03
JEMP IS REPLACED BY EMPLOYEE NUMBER
HRS IS REPLACED BY HOURS WORKED
IDEP IS REPLACED BY DEPARTMENT NUMBER
IEMP IS REPLACED BY EMPLOYEE NUMBER
INM IS REPLACED BY EMPLOYEE NAME
ISS IS REPLACED BY SOCIAL SECURITY NUMBER
IRT IS REPLACED BY PAY RATE
IYTD IS REPLACED BY YEAR-TO-DATE GROSS

Sample Problem 3: Source Program

```
// JOB                                                                  CSP31510
// FOR                                                                  CSP31520
* NAME SP3                                                             CSP31530
*IOCS(CARD,1132 PRINTER,TYPEWRITER)                                    CSP31540
* ONE WORD INTEGERS                                                    CSP31550
* EXTENDED PRECISION                                                   CSP31560
* LIST ALL                                                            CSP31570
      DIMENSION MASK(12),IN(69),IDEP(2),IEMP(3),INM(20),ISS(9),IRT(4),  CSP31580
     1 IYTD(7),JEMP(3),NYTD(7),ICUR(6),KCURR(12),KOYTD(12),KNYTD(12)    CSP31590
    1 FORMAT (69A1,I1)                                                  CSP31600
    2 FORMAT (12A1)                                                     CSP31610
20    FORMAT (1H ,2A1,1X,23A1,2X,20A1,21X,1H1,3X,7HCSP     )            CSP31620
30    FORMAT (1H ,2A1,2X,3A1,2X,20A1,5X,3(12A1,2X))                     CSP31630
      CALL OATSW(0,I)                                                   CSP31640
      CALL DATSW(1,M)                                                   CSP31650
      CALL OATSW(2,L)                                                   CSP31660
      NREAD=6*(1/L)+2                                                   CSP31670
      NWRIT=2*(1/I)+2*(1/M)+1                                           CSP31680
      READ (NREAD,2) MASK                                               CSP31690
   15 READ (NREAD,1) IN,ICD                                             CSP31700
      IF (ICD) 6,10,6                                                   CSP31710
6     NZERO=0                                                           CSP31720
      GO TO (7,8), ICD                                                  CSP31730
C     THIS IS THE YEAR TO DATE PROCESSING                               CSP31740
    7 CALL MOVE (IN,1,2,IDEP,1)                                         CSP31750
      CALL MOVE (IN,4,6,IEMP,1)                                         CSP31760
      CALL MOVE (IN,7,26,INM,1)                                         CSP31770
      CALL MOVE (IN,29,37,ISS,1)                                        CSP31780
      CALL MOVE (IN,38,41,IRT,1)                                        CSP31790
      CALL MOVE (IN,42,48,IYTD,1)                                       CSP31800
      GO TO 15                                                          CSP31810
C     THIS IS CURRENT PERIOD PROCESSING                                 CSP31820
    8 CALL MOVE (IN,1,3,JEMP,1)                                         CSP31830
      HRS=GET (IN,28,30,100,0)                                          CSP31840
      GO TO 15                                                          CSP31850
10    NZERO = NZERO + 1                                                 CSP31860
      IF (NZERO - 1) 100,100,101                                        CSP31870
101   STOP 3333                                                         CSP31880
100   IF (NCOMP(IEMP,1,3,JEMP,1)) 99,11,99                             CSP31890
   11 CURR=(HRS*GET(IRT,1,4,10,0)+500,0)/1000,0                         CSP31900
      YTD=CURR+GET (IYTD,1,7,10,0)                                      CSP31910
      CALL PUT (NYTO,1,7,YTD,5,0,1)                                     CSP31920
      WRITE (1,20) IDEP,IEMP,INM,ISS,IRT,NYTO                           CSP31930
      CALL PUT (ICUR,1,6,CURR,5,0,1)                                    CSP31940
      CALL MOVE (MASK,1,12,KCURR,1)                                     CSP31950
      CALL MOVE (MASK,1,12,KOYTD,1)                                     CSP31960
      CALL MOVE (MASK,1,12,KNYTO,1)                                     CSP31970
      CALL EDIT (ICUR,1,6,KCURR,1,12)                                   CSP31980
      CALL EDIT (IYTO,1,7,KOYTO,1,12)                                   CSP31990
      CALL EDIT (NYTD,1,7,KNYTO,1,12)                                   CSP32000
      WRITE (NWRIT,30) IDEP,IEMP,INM,KOYTD,KCURR,KNYTD                  CSP32010
      GO TO 15                                                          CSP32020
C     THIS IS AN ERROR.  THE EMP NOS DO NOT MATCH.                      CSP32030
99    WRITE (1,40)                                                      CSP32040
40    FORMAT (' THE EMP NOS DO NOT MATCH.')                             CSP32050
      GO TO 15                                                          CSP32060
```

```
 SAMPLE PROBLEM 3                                                  PAGE 02
      END                                                          CSP32070

VARIABLE ALLOCATIONS
 HRS  =0000  CURR =0003  YTD  =0006  MASK =0017  IN   =005C  IDEP =005E  IEMP =0061  INM  =0075  ISS  =007E  IRT  =0082
 IYTO =0089  JEMP =008C  NYTD =0093  ICUR =0099  KCURR=00A5  KOYTD=00B1  KNYTD=00BD  I    =00BE  M    =00BF  L    =00C0
 NREAO=00C1  NWRIT=00C2  ICD  =00C3  NZERO=00C4

STATEMENT ALLOCATIONS
 1    =00E8  2    =00EC  20   =00EF  30   =0103  40   =0114  15   =016C  6    =0178  7    =0182  8    =01AE  10   =01BF
 101  =01CB  100  =01C0  11   =0106  99   =0259

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION
 IOCS

CALLED SUBPROGRAMS
 DATSW   MOVE    GET     NCOMP   PUT     EOIT    EADO    EMPY    EOIV    ELO     ESTO    WRTYZ   SREO    SWRT    SCOMP
 SFIO    SIOAI   SIOI    STOP    CAROZ   PRNTZ

REAL CONSTANTS
   .100000000E 03=00C6     .100000000E 02=00C9     .500000000E 03=00CC     .100000000E 04=00CF     .500000000E 01=00D2

INTEGER CONSTANTS
     0=00D5      1=00D6      2=00D7      6=00D8      4=00D9      7=00DA     26=00DB     29=00DC     37=0000     38=00DE
    41=00DF     42=00E0     48=00E1      3=00E2     28=00E3     30=00E4   3333=00E5     12=00E6  13107=00E7

CORE REQUIREMENTS FOR SP3
 COMMON       0  VARIABLES    198  PROGRAM    410

END OF COMPILATION
```

Sample Problem 3: Payroll Register Output

```
// XEQ                                                          CSP32080

01  101  NALNIUQ , J              $7,453.06      $198.91      $7,651.97
52  201  OMINOREG,     M         $3,524.37      $143.82      $3,668.19
76  676  NEDAB,    R             $10,060.60     $297.27      $10,357.87
76  689  NEDUOL,   R             $10,060.60     $297.27      $10,357.87
01  253  NROH , J                $9,555.62      $279.65      $9,835.27
```

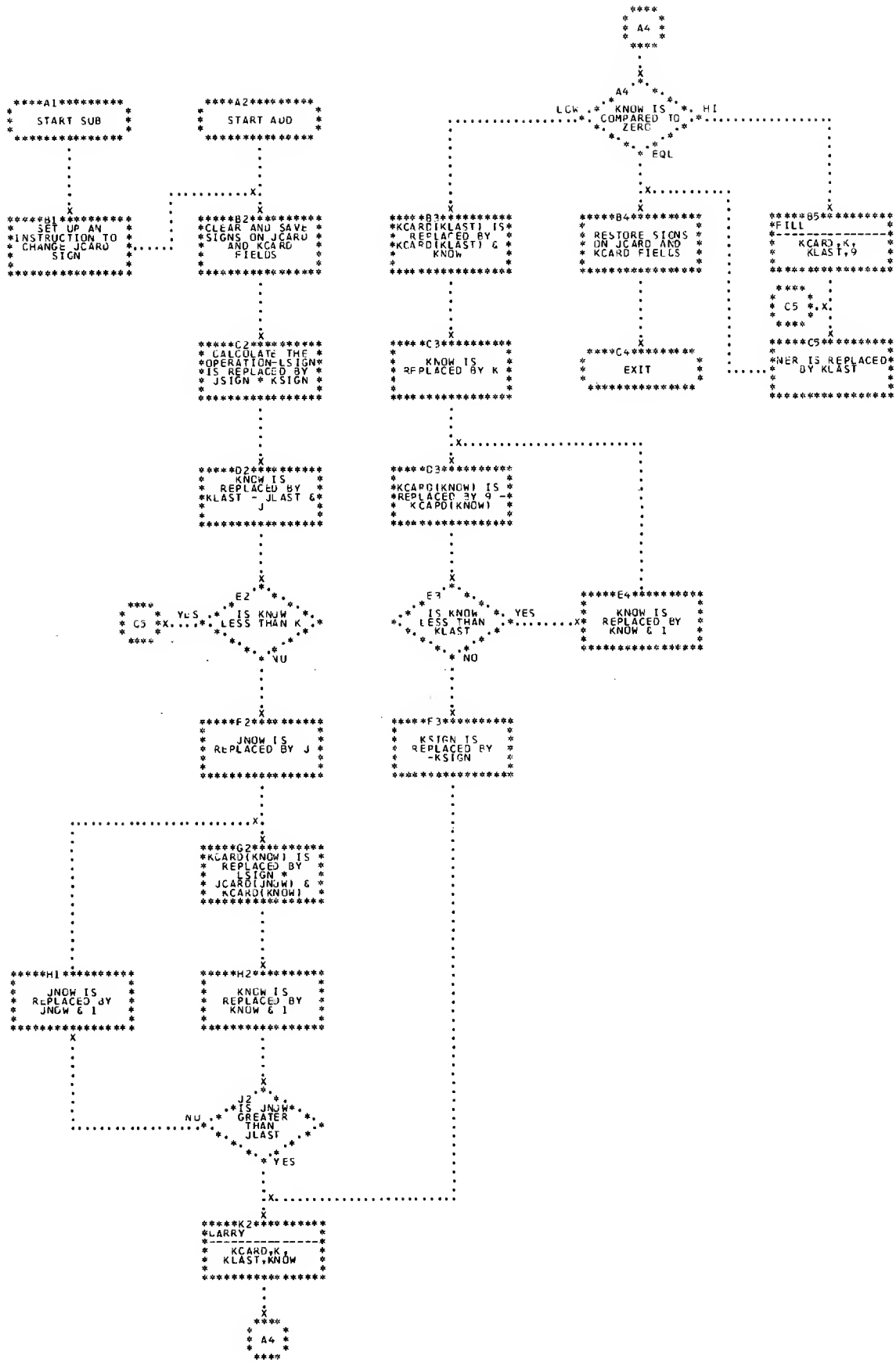
Sample Problem 3: Console Printer Error Log and New Year-to-Date Card Image

```
01 101NALNIUQ,  J          7985664320542076519 7     1   CSP

52 2010MINOREG,    M       0133256780423036681 9     1   CSP

76 676NEDAB,    R          0142330600810103578 7     1   CSP

76 689NEDUOL,   R          7986037940810103578 7     1   CSP

THE EMP NOS DO NOT MATCH.

01 253NROH,     J          9546230570762098352 7     1   CSP
```

Sample Problem 3: Data Input Listing

```
// XEQ                                                                  CSP32080
   ,  $,  CR                                                            CSP32090
01 101NALNIUQ , J          79856643205420745306                      1  CSP32100
101NALNIUQ , J          01367                                        2  CSP32110
                                                                     0  CSP32120
2010MINOREG,     M      52340                                        2  CSP32130
52 2010MINOREG,     M      01332567804230352437                      1  CSP32140
                                                                     0  CSP32150
76 676NEDAB,    R          01423306008101006060                      1  CSP32160
676NEOAB,    R          76367                                        2  CSP32170
                                                                     0  CSP32180
689NEOUOL,    R         76367                                        2  CSP32190
76 689NEOUOL,    R         79860379408101006060                      1  CSP32200
                                                                     0  CSP32210
99 9990NATNOM J            99999999901160511122                      1  CSP32220
0990NATNOM , J          994009                                       2  CSP32230
                                                                     0  CSP32240
01 253NROH , J             95462305707620955562                      1  CSP32250
253NROH , J             01367                                        2  CSP32260
                                                                     0  CSP32270
                                                                        CSP32280
```

CHART AD 1130 COMMERCIAL ADD/SUB SUBROUTINE



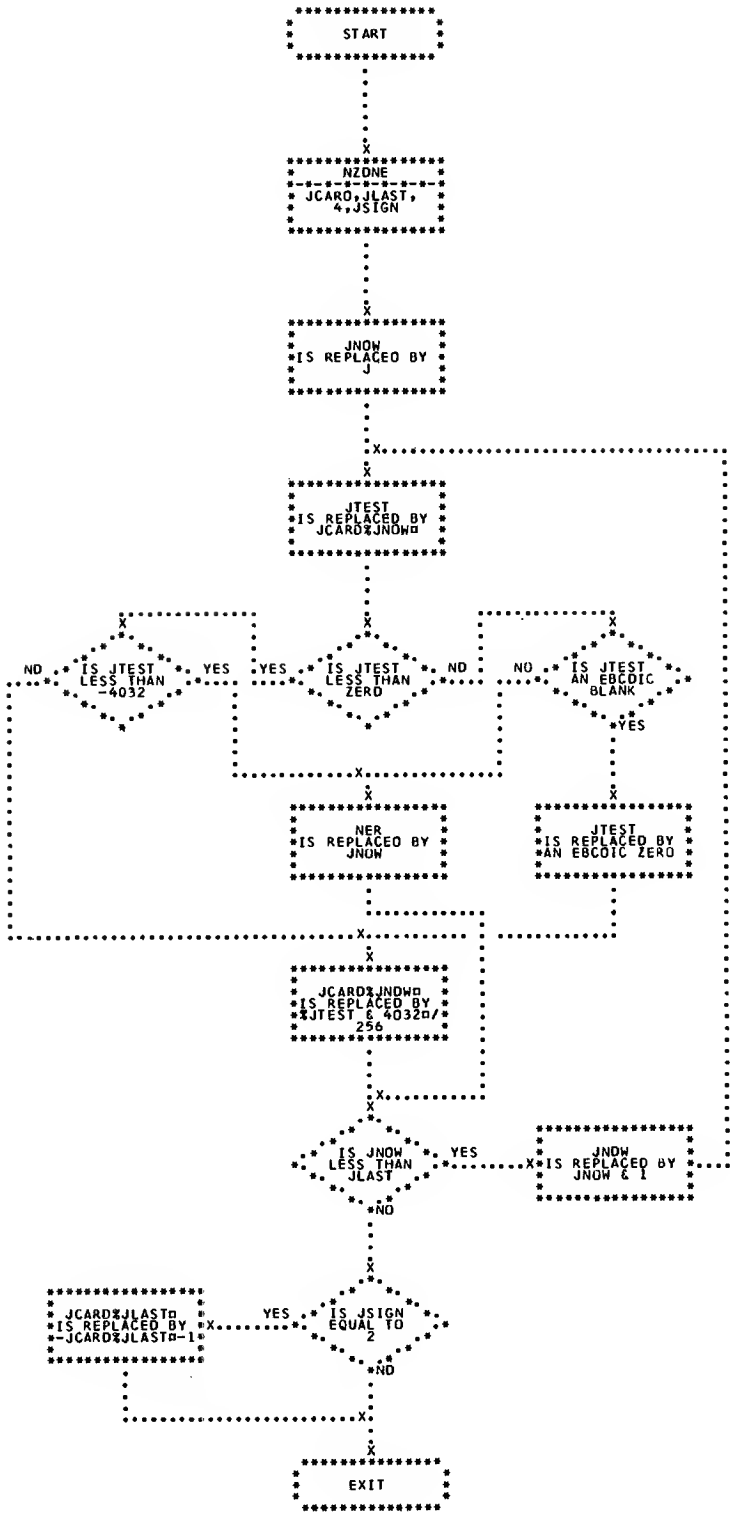
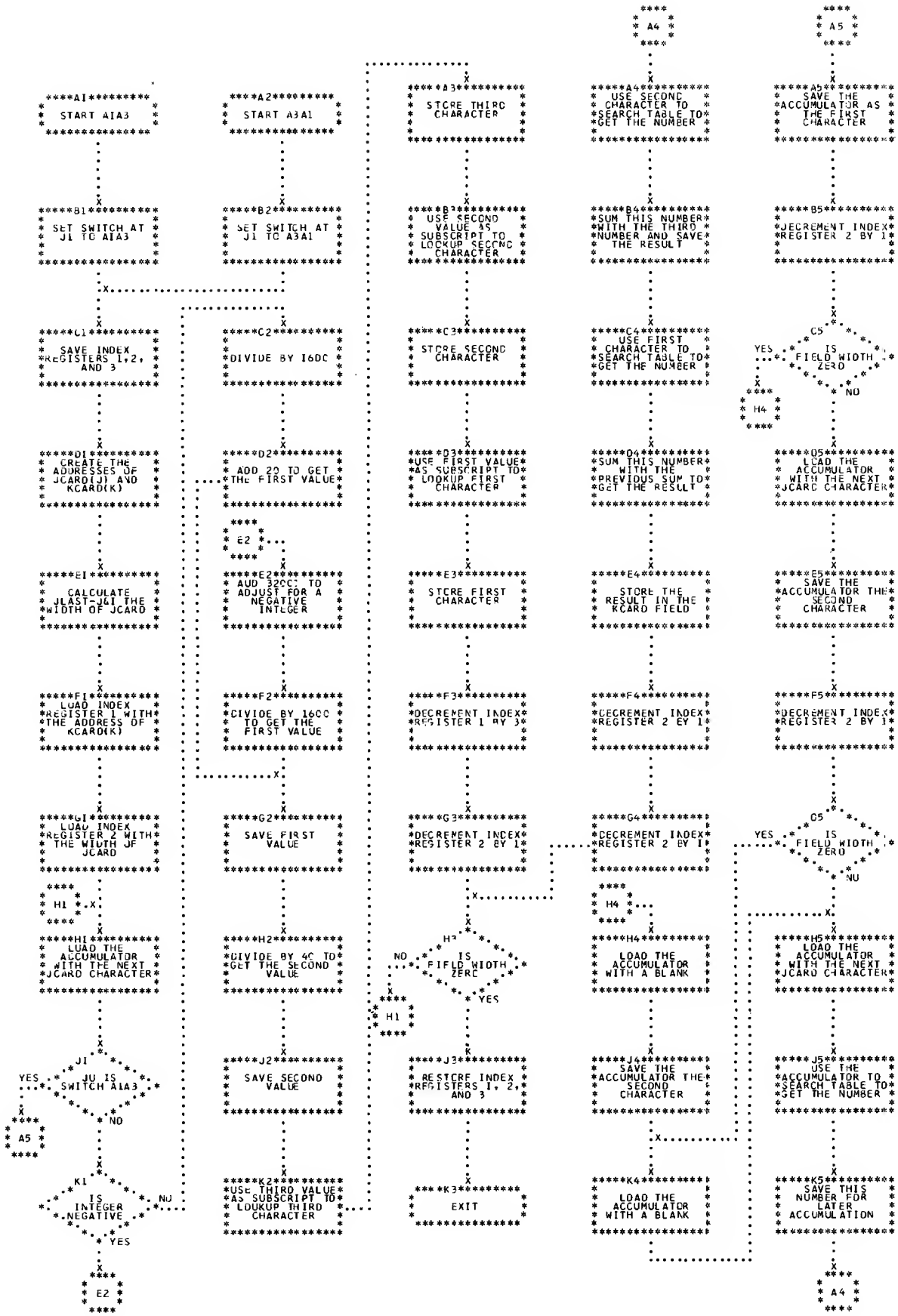

START
NZONE
JCARD,JLAST,
4,JSIGN
JNOW
IS REPLACED BY
J
JTEST
IS REPLACED BY
JCARD%JNOW□
IS JTEST
LESS THAN
-4032
NO
YES
IS JTEST
LESS THAN
ZERO
YES
NO
IS JTEST
AN EBCDIC
BLANK
NO
YES
NER
IS REPLACED BY
JNOW
JTEST
IS REPLACED BY
AN EBCDIC ZERO
JCARD%JNDW□
IS REPLACED BY
%JTEST & 4032□/
256
IS JNOW
LESS THAN
JLAST
YES
NO
JNDW
IS REPLACED BY
JNOW & 1
JCARD%JLAST□
IS REPLACED BY
-JCARD%JLAST□-1
YES
IS JSIGN
EQUAL TO
2
NO
EXIT

CHART A3 1130 COMMERCIAL A1A3 SUBROUTINE



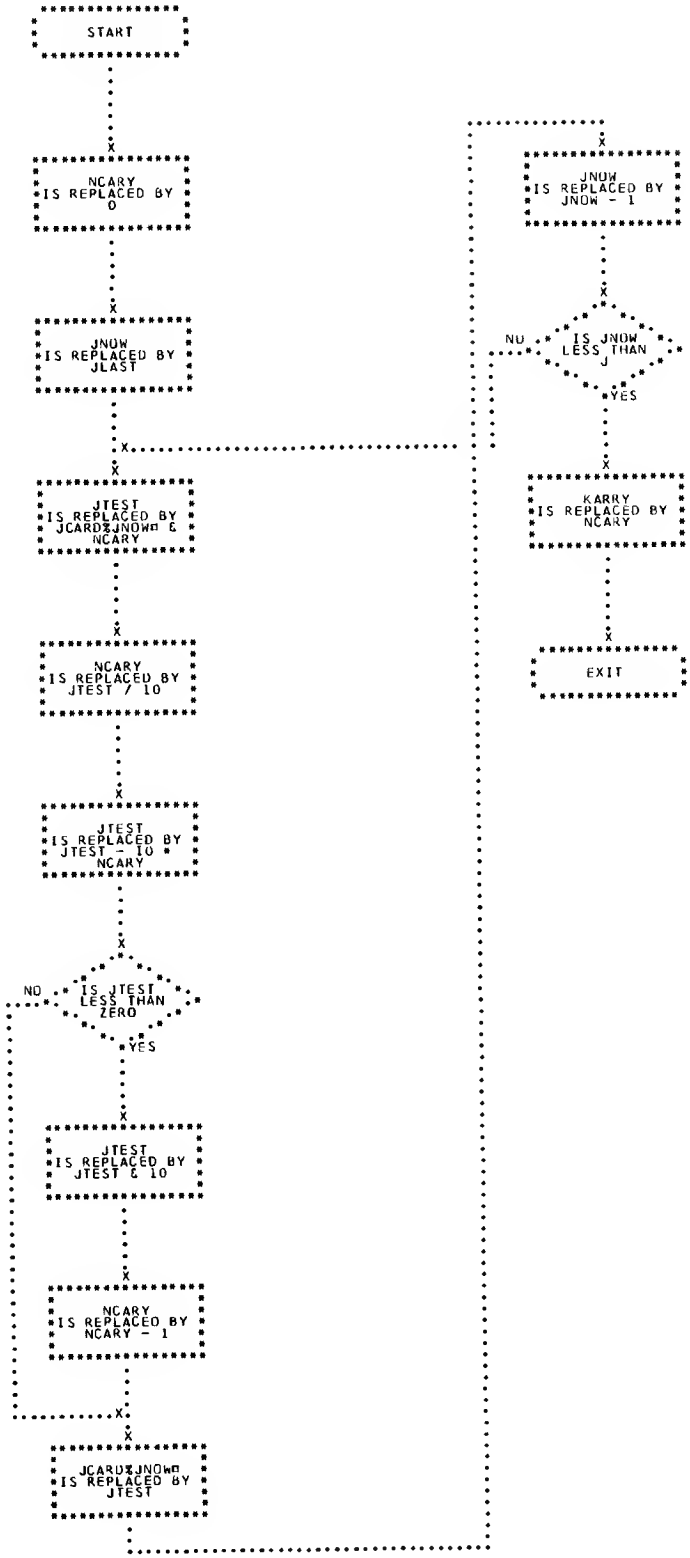
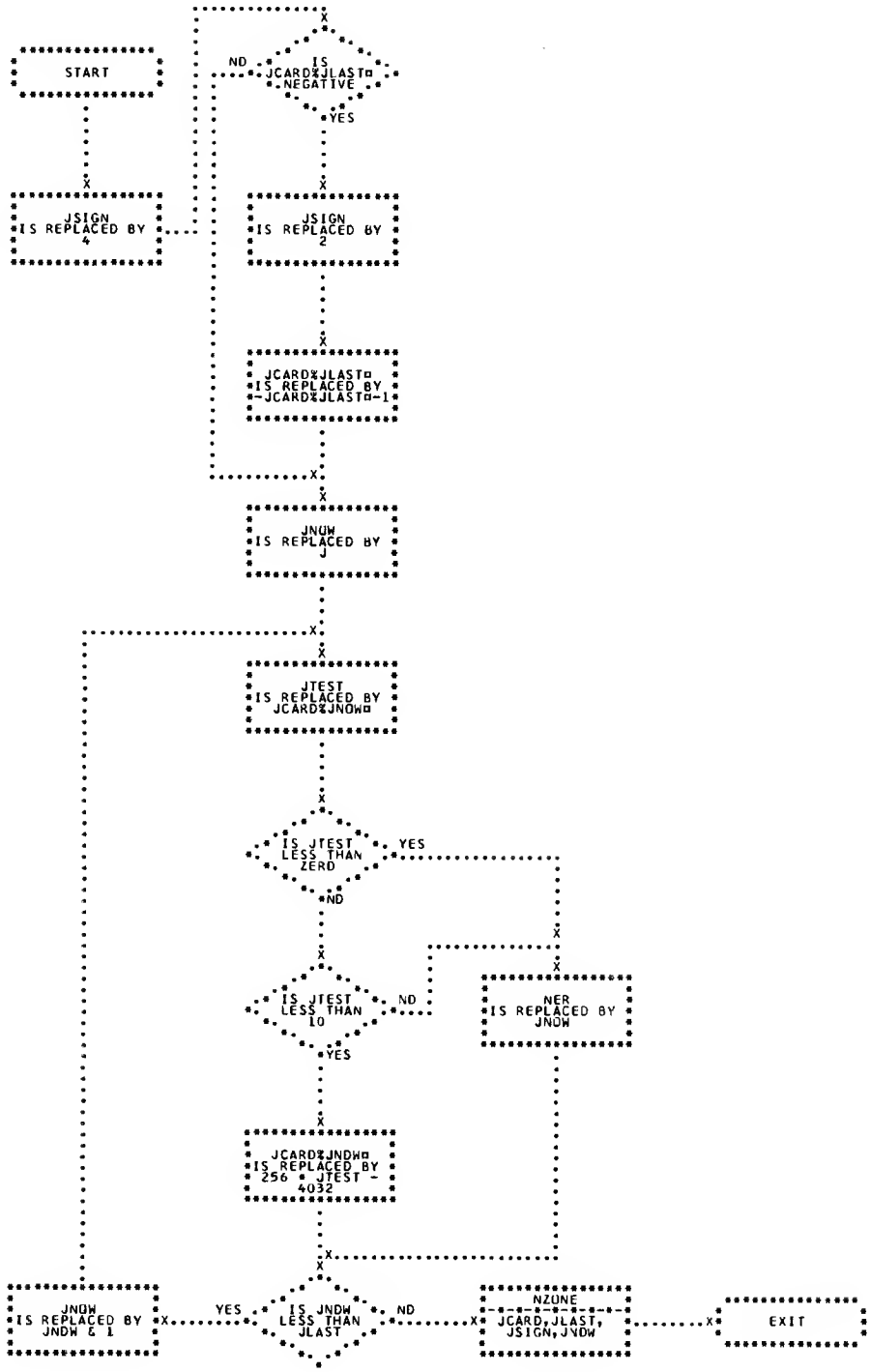
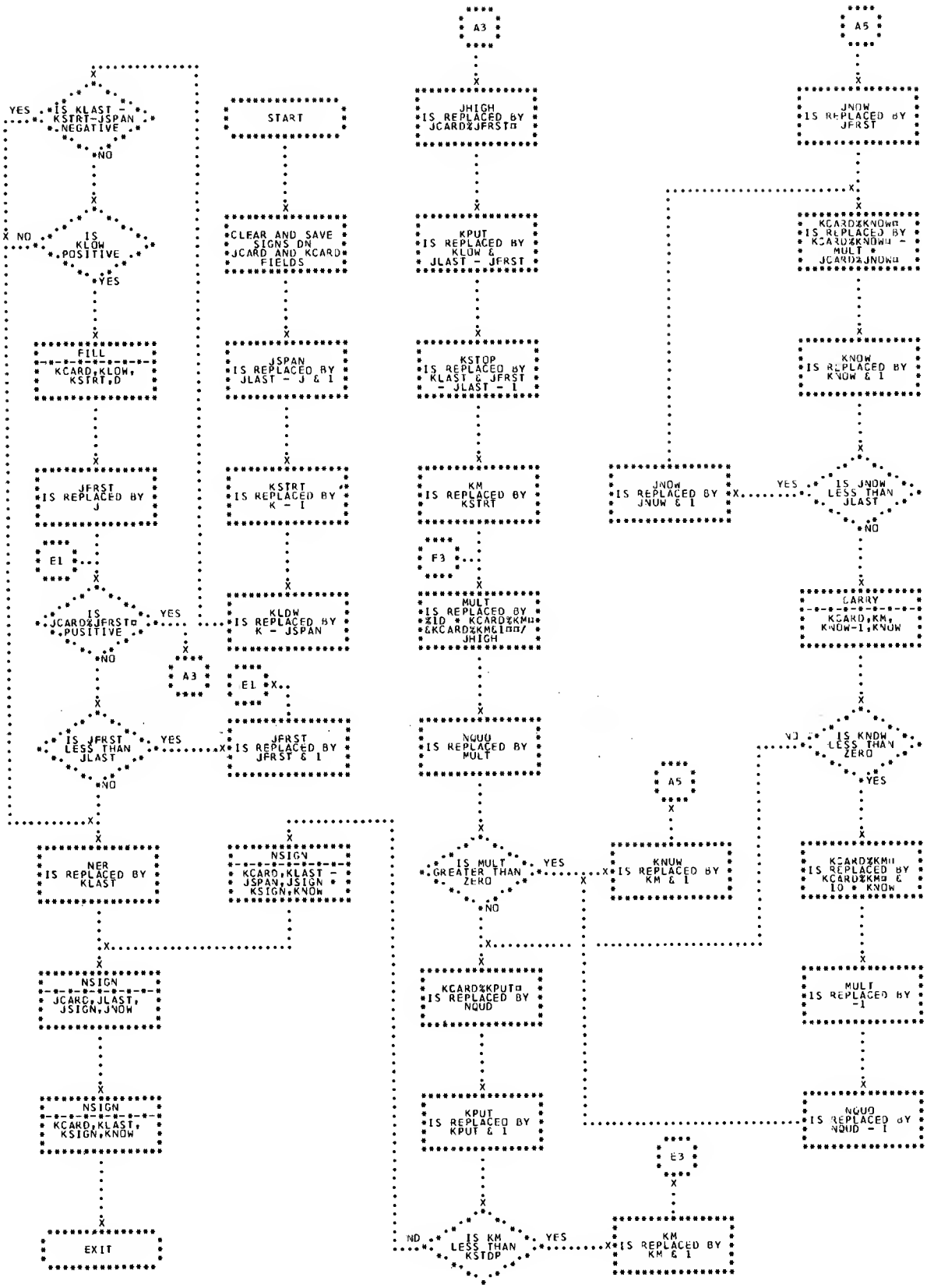
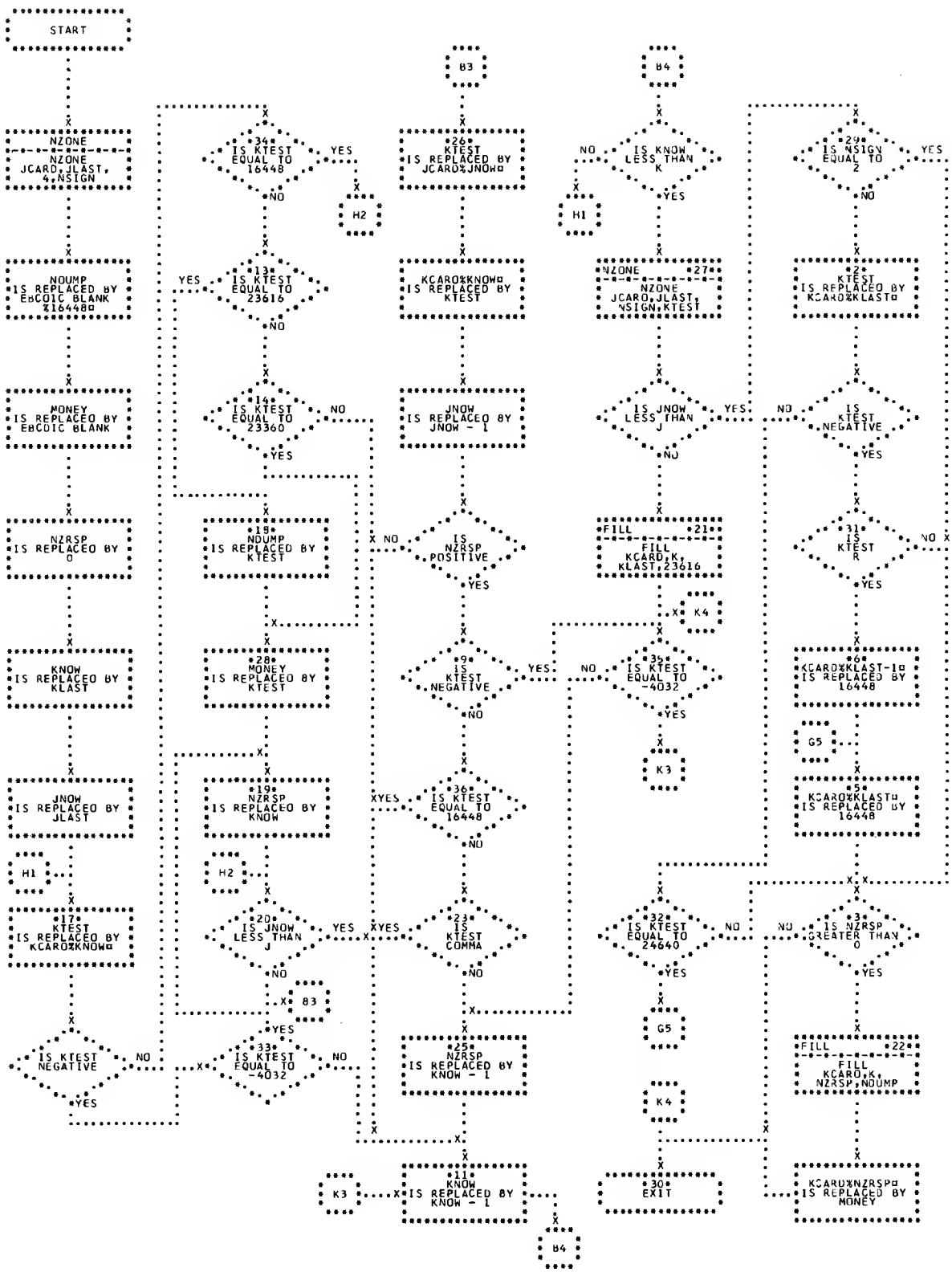
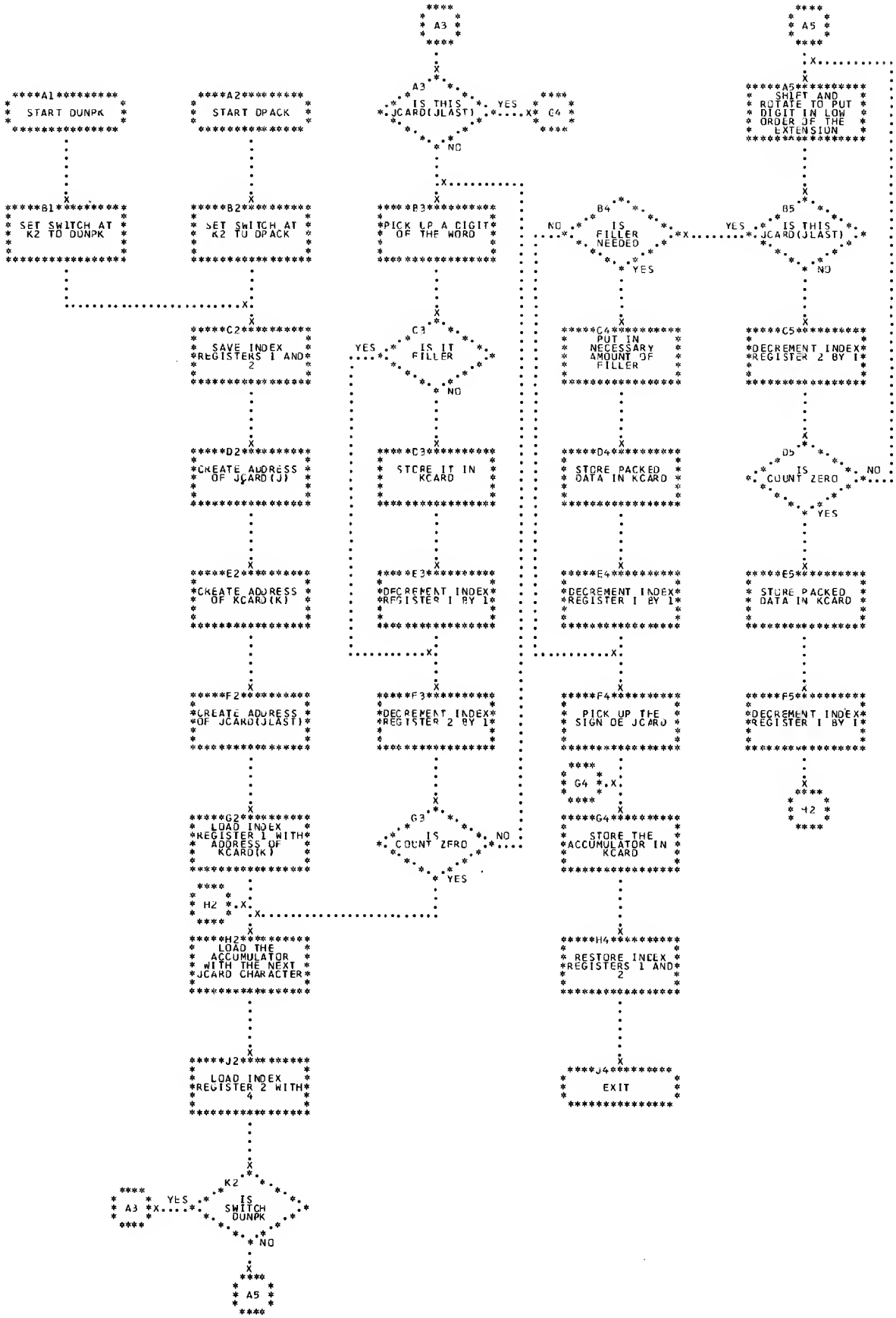

START
NCARY IS REPLACED BY 0
JNOW IS REPLACED BY JLAST
JTEST IS REPLACED BY JCARD&JNOW□ & NCARY
NCARY IS REPLACED BY JTEST / 10
JTEST IS REPLACED BY JTEST - 10 * NCARY
IS JTEST LESS THAN ZERO
NO
YES
JTEST IS REPLACED BY JTEST & 10
NCARY IS REPLACED BY NCARY - 1
JCARD&JNOW□ IS REPLACED BY JTEST
JNOW IS REPLACED BY JNOW - 1
IS JNOW LESS THAN J
NO
YES
KARRY IS REPLACED BY NCARY
EXIT

CHART DE 1130 COMMERCIAL DECA1 SUBROUTINE

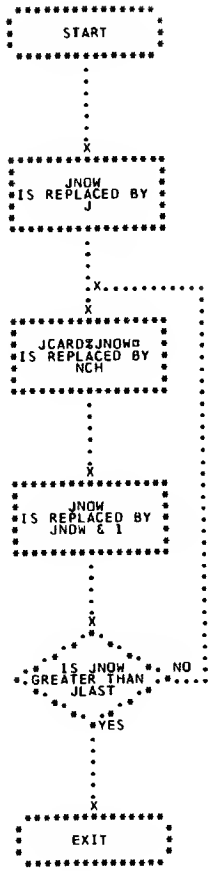


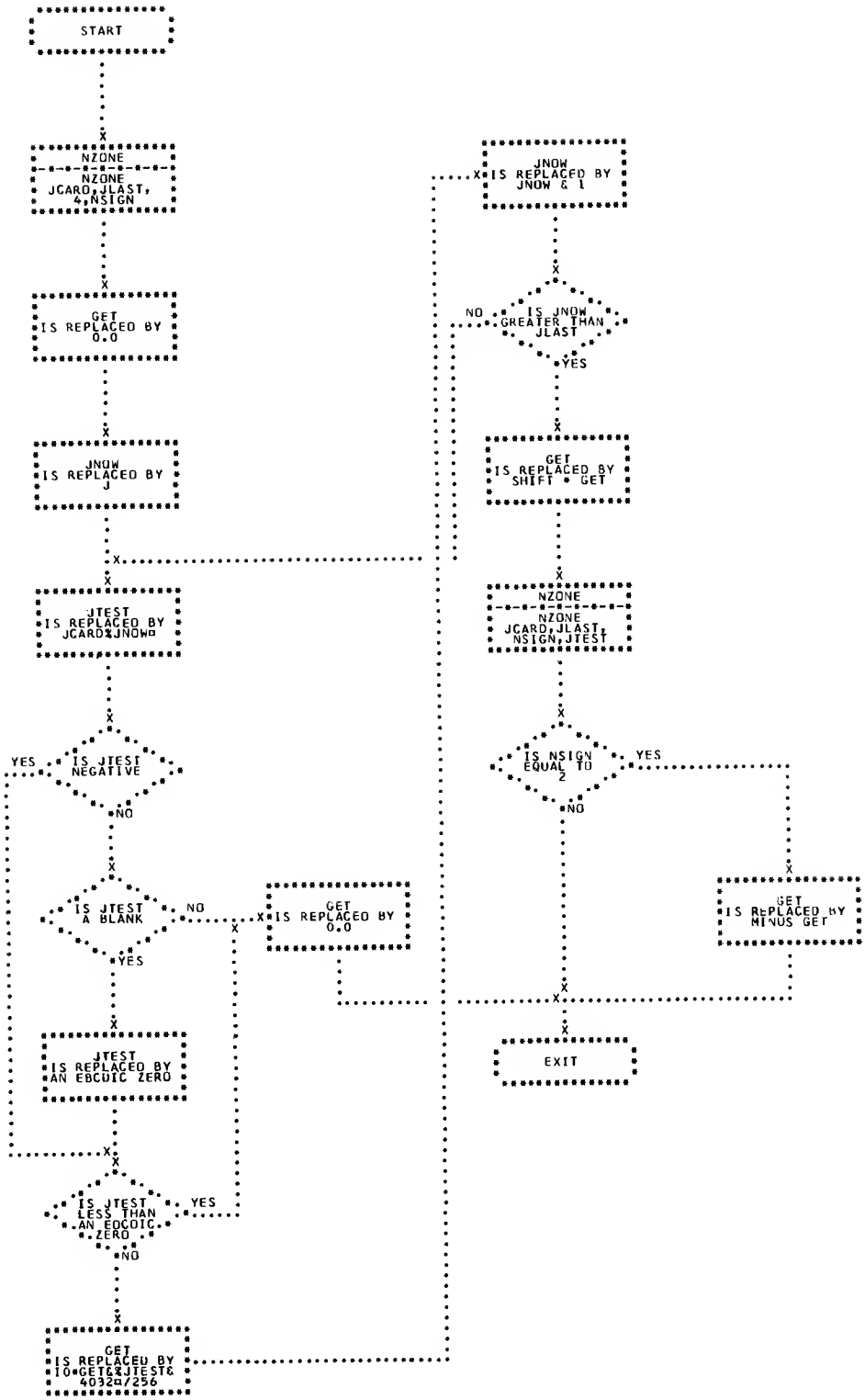

IS KLAST - KSTRT-JSPAN NEGATIVE
YES
NO
IS KLOW POSITIVE
NO
YES
FILL KCARD,KLOW, KSTRT,D
JFRST IS REPLACED BY J
E1
IS JCARD&JFRST¤ PUSITIVE
YES
NO
A3
IS JFRST LESS THAN JLAST
YES
NO
NER IS REPLACED BY KLAST
NSIGN JCARD,JLAST, JSIGN,JNOW
NSIGN KCARD,KLAST, KSIGN,KNOW
EXIT
START
CLEAR AND SAVE SIGNS ON JCARD AND KCARD FIELDS
JSPAN IS REPLACED BY JLAST - J & 1
KSTRT IS REPLACED BY K - I
KLOW IS REPLACED BY K - JSPAN
A3
E1
JFRST IS REPLACED BY JFRST & 1
NSIGN KCARD,KLAST - JSPAN,JSIGN KSIGN,KNOW
A3
JHIGH IS REPLACED BY JCARD&JFRST¤
KPUT IS REPLACED BY KLOW & JLAST - JFRST
KSTOP IS REPLACED BY KLAST & JFRST - JLAST - 1
KM IS REPLACED BY KSTRT
F3
MULT IS REPLACED BY ¤ID * KCARD&KM¤ &KCARD&KM&1¤¤/ JHIGH
NQUO IS REPLACED BY MULT
IS MULT GREATER THAN ZERO
YES
NO
KCARD&KPUT¤ IS REPLACED BY NQUD
KPUT IS REPLACED BY KPUT & 1
IS KM LESS THAN KSTOP
ND
YES
A5
JNOW IS REPLACED BY JFRST
KCARD&KNOW¤ IS REPLACED BY KCARD&KNOW¤ - MULT * JCARD&JNOW¤
KNOW IS REPLACED BY KNOW & 1
IS JNOW LESS THAN JLAST
YES
NO
CARRY KCARD,KM, KNOW-1,KNOW
IS KNOW LESS THAN ZERO
YES
KCARD&KM¤ IS REPLACED BY KCARD&KM¤ & 10 * KNOW
MULT IS REPLACED BY -1
NQUO IS REPLACED BY NQUD - 1
A5
JNOW IS REPLACED BY JNOW & 1
KNOW IS REPLACED BY KM & 1
E3
KM IS REPLACED BY KM & 1

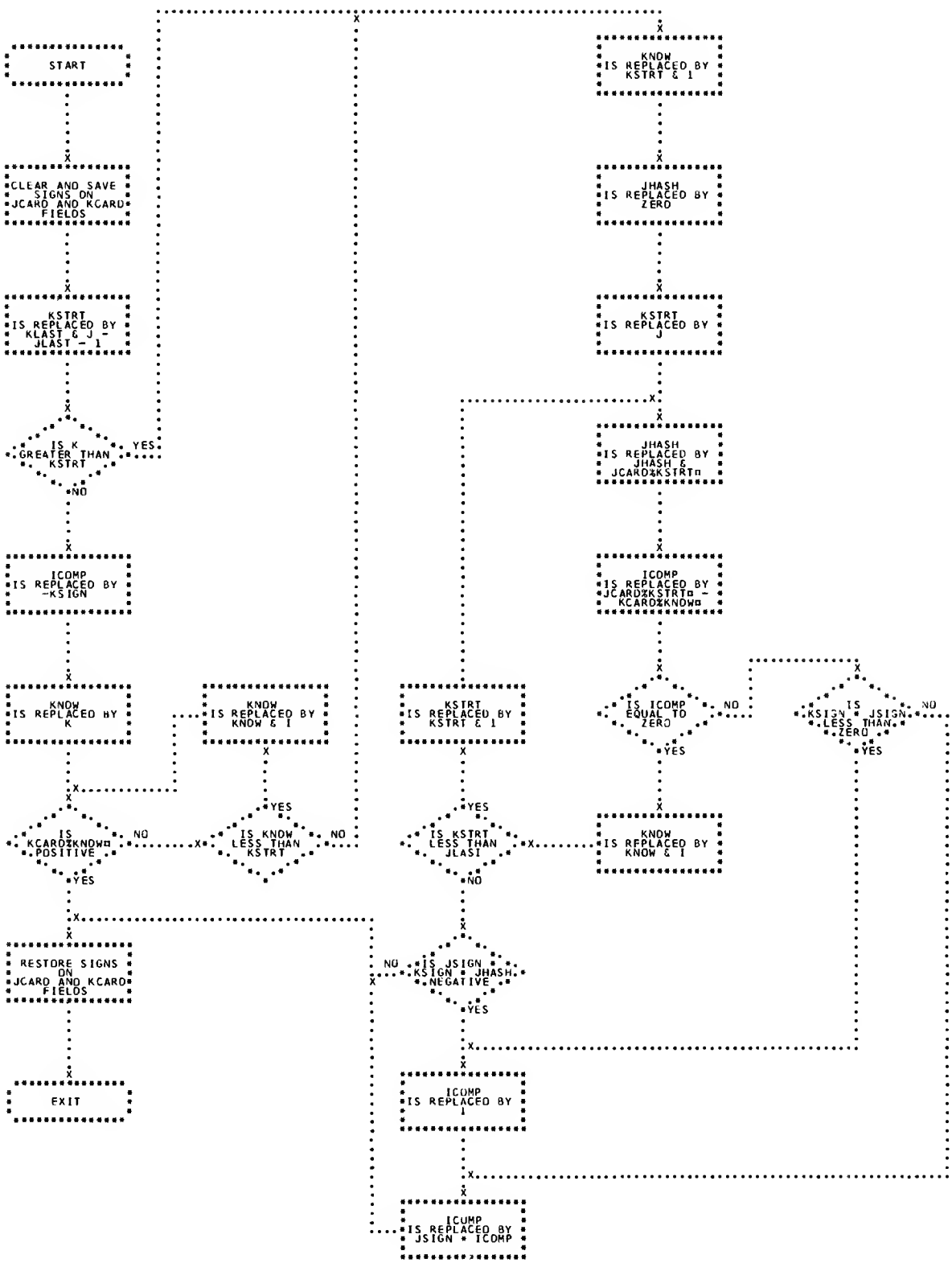

START
NZONE
NZONE JCARD,JLAST, 4,NSIGN
NOUMP IS REPLACED BY EBCDIC BLANK %16448¤
MONEY IS REPLACED BY EBCDIC BLANK
NZRSP IS REPLACED BY 0
KNOW IS REPLACED BY KLAST
JNOW IS REPLACED BY JLAST
H1
17 KTEST IS REPLACED BY KCARD%KNOW¤
IS KTEST NEGATIVE
NO
YES
34 IS KTEST EQUAL TO 16448
YES
NO
H2
13 IS KTEST EQUAL TO 23616
YES
NO
14 IS KTEST EQUAL TO 23360
NO
YES
18 NOUMP IS REPLACED BY KTEST
28 MONEY IS REPLACED BY KTEST
19 NZRSP IS REPLACED BY KNOW
H2
20 IS JNOW LESS THAN J
YES
NO
X 83
83
YES
33 IS KTEST EQUAL TO -4032
NO
K3
B3
26 KTEST IS REPLACED BY JCARD%JNOW¤
KCARD%KNOW¤ IS REPLACED BY KTEST
JNOW IS REPLACED BY JNOW - 1
IS NZRSP POSITIVE
NO
YES
9 IS KTEST NEGATIVE
YES
NO
36 IS KTEST EQUAL TO 16448
YES
NO
23 IS KTEST COMMA
YES
NO
25 NZRSP IS REPLACED BY KNOW - 1
11 KNOW IS REPLACED BY KNOW - 1
B4
B4
IS KNOW LESS THAN K
NO
YES
H1
27 NZONE NZONE JCARD,JLAST, NSIGN,KTEST
IS JNOW LESS THAN J
YES
NO
21 FILL FILL KCARD,K, KLAST,23616
K4
35 IS KTEST EQUAL TO -4032
NO
YES
K3
32 IS KTEST EQUAL TO 24640
NO
YES
G5
K4
30 EXIT
29 IS NSIGN EQUAL TO 2
YES
NO
2 KTEST IS REPLACED BY KCARD%KLAST¤
IS KTEST NEGATIVE
NO
YES
31 IS KTEST R
NO
YES
6 KCARD%KLAST-1¤ IS REPLACED BY 16448
G5
5 KCARD%KLAST¤ IS REPLACED BY 16448
3 IS NZRSP GREATER THAN 0
NO
YES
22 FILL FILL KCARD,K, NZRSP,NOUMP
KCARD%NZRSP¤ IS REPLACED BY MONEY

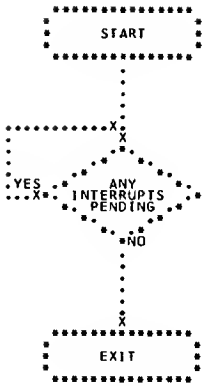


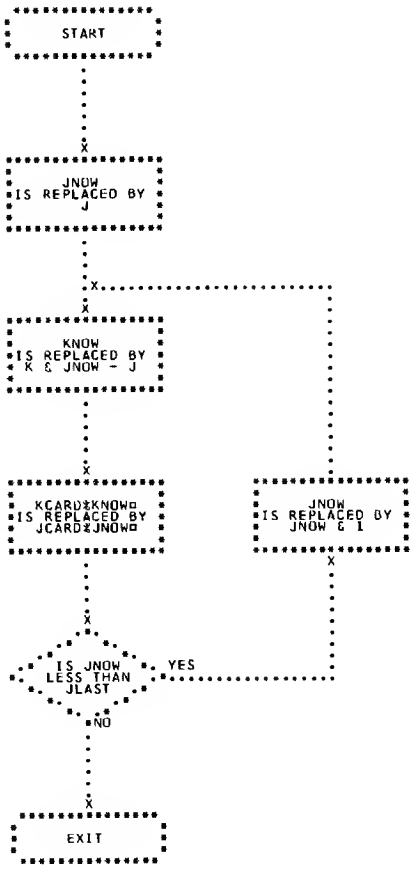
ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

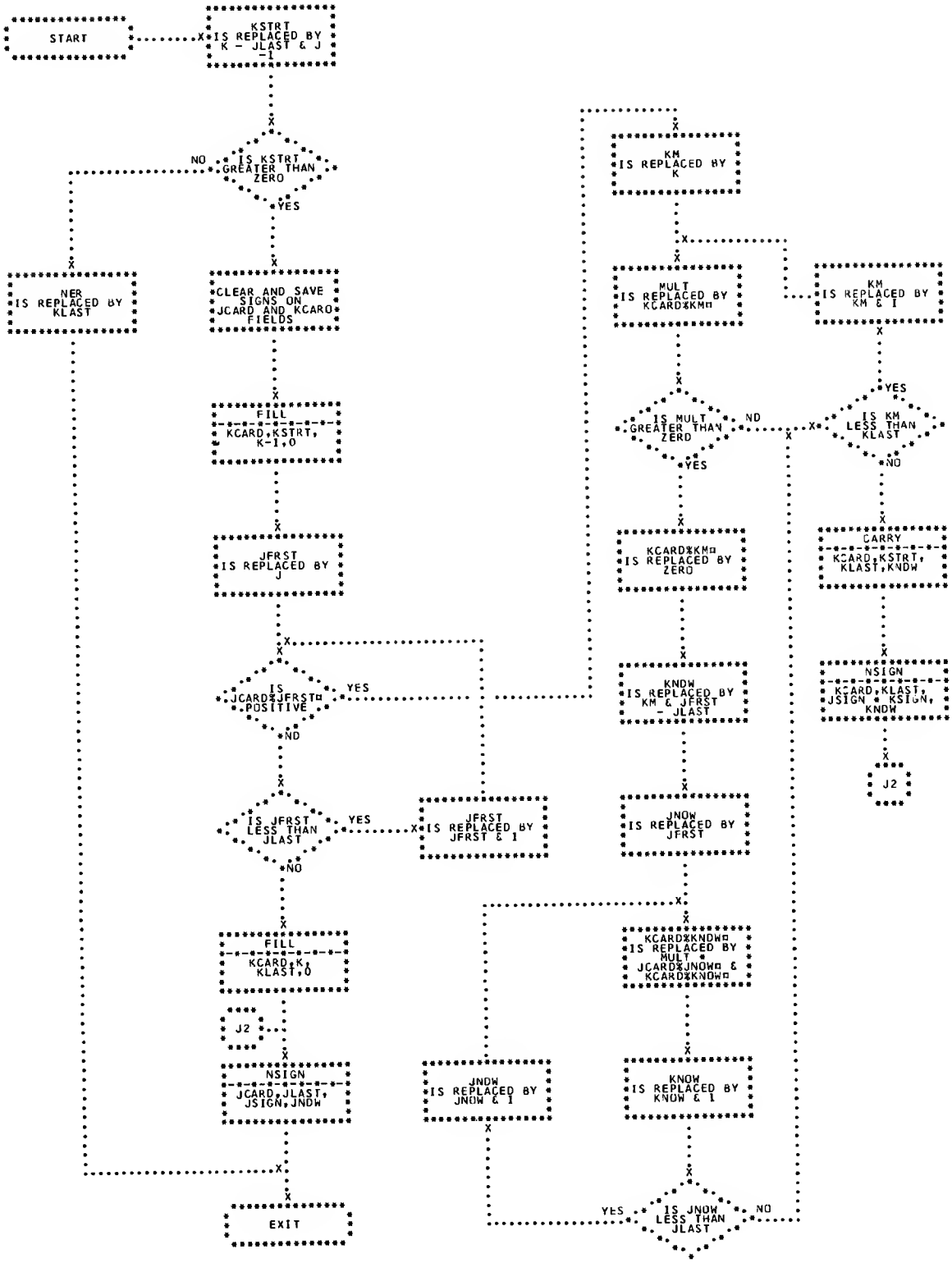



START
NZONE
NZONE JCARD,JLAST, 4,NSIGN
GET IS REPLACED BY 0.0
JNOW IS REPLACED BY J
JTEST IS REPLACED BY JCARD%JNOW¤
IS JTEST NEGATIVE
YES
NO
IS JTEST A BLANK
NO
YES
GET IS REPLACED BY 0.0
JTEST IS REPLACED BY AN EBCDIC ZERO
IS JTEST LESS THAN AN EBCDIC ZERO
YES
NO
GET IS REPLACED BY IO*GET&%JTEST& 4032¤/256
JNOW IS REPLACED BY JNOW & 1
IS JNOW GREATER THAN JLAST
NO
YES
GET IS REPLACED BY SHIFT * GET
NZONE
NZONE JCARD,JLAST, NSIGN,JTEST
IS NSIGN EQUAL TO 2
YES
NO
GET IS REPLACED BY MINUS GET
EXIT



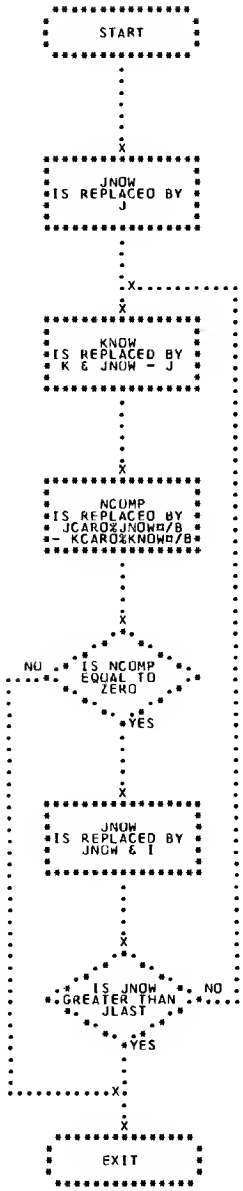






ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

CHART CO I130 COMMERCIAL NCOMP FUNCTION



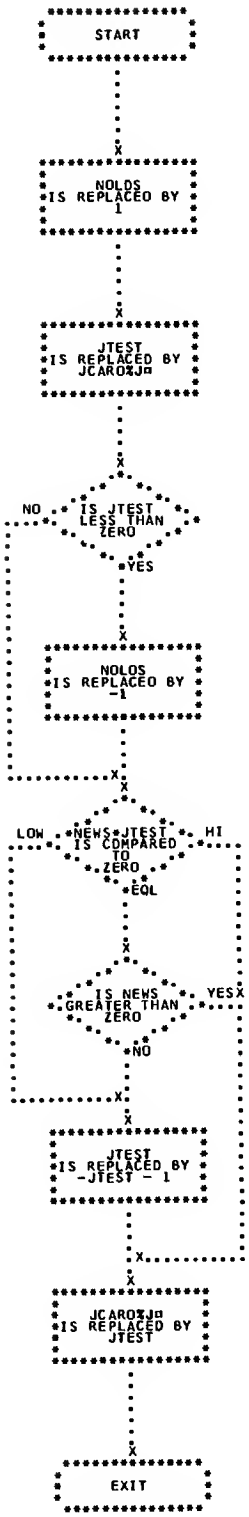
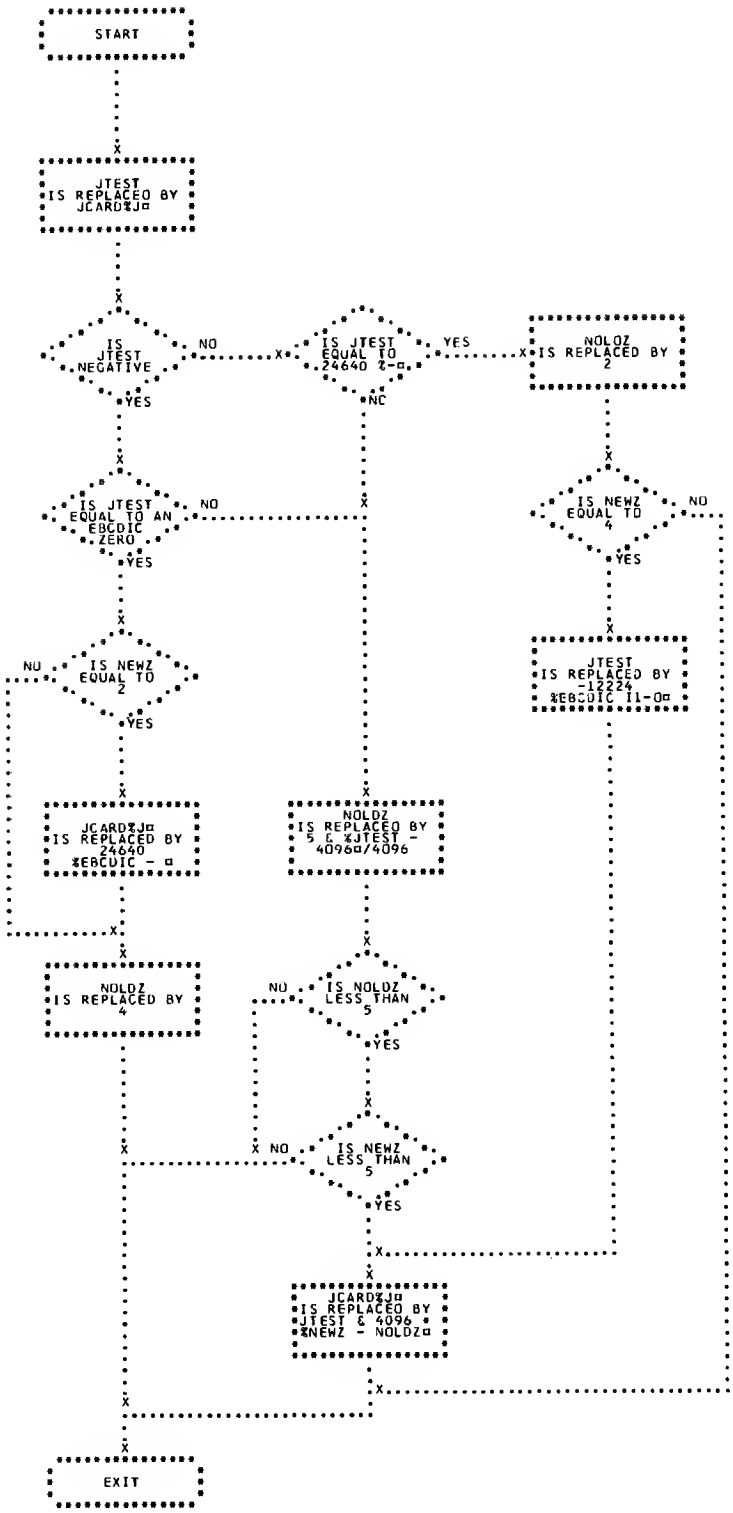
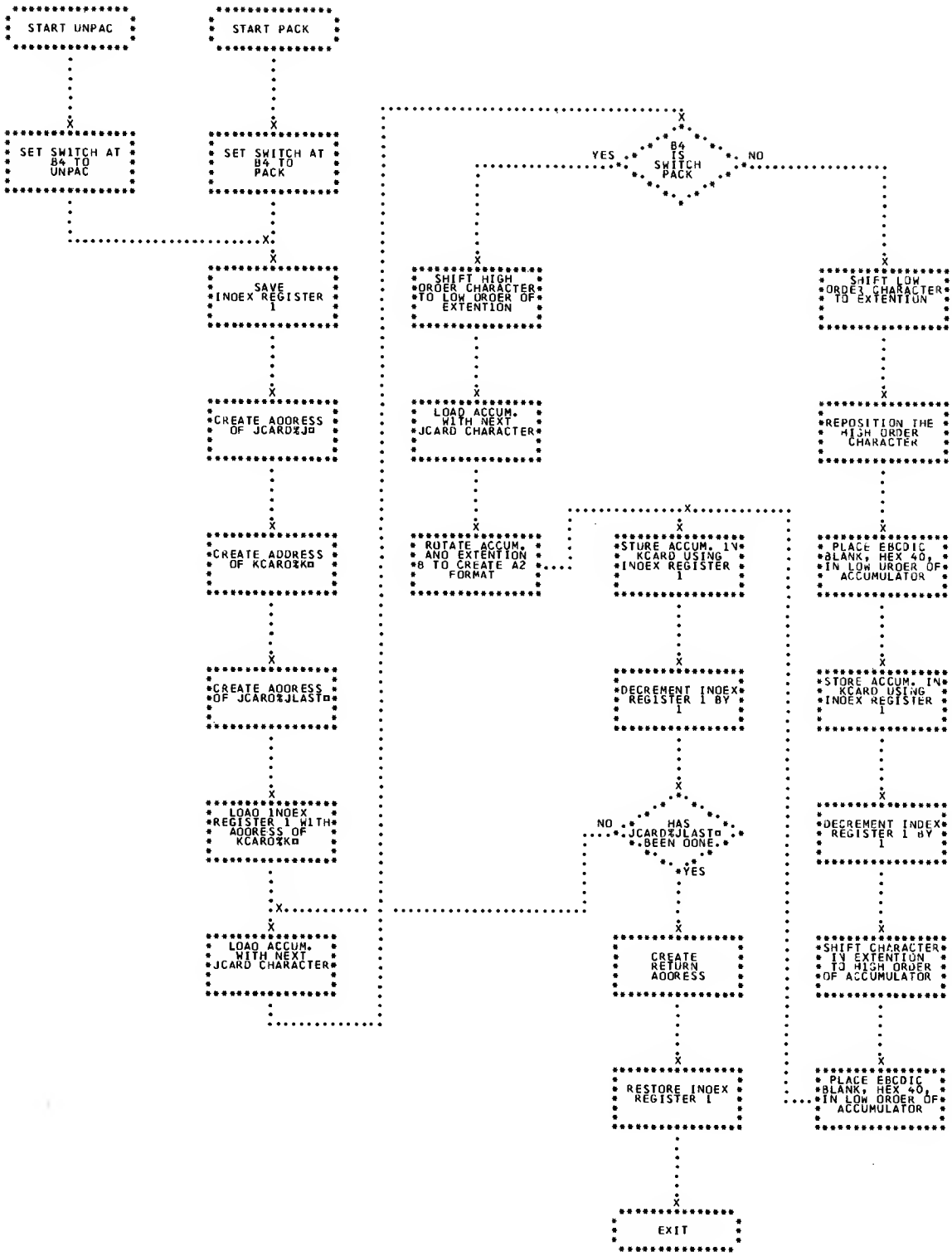
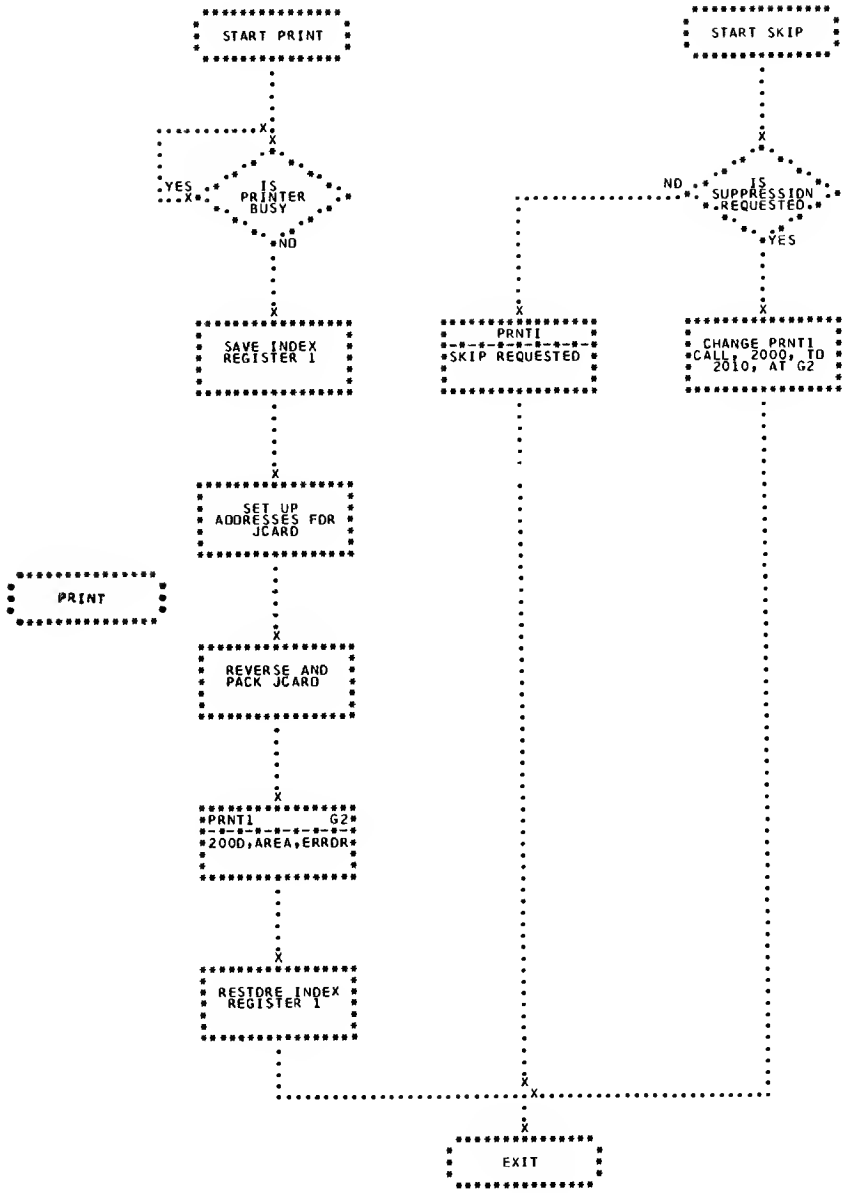
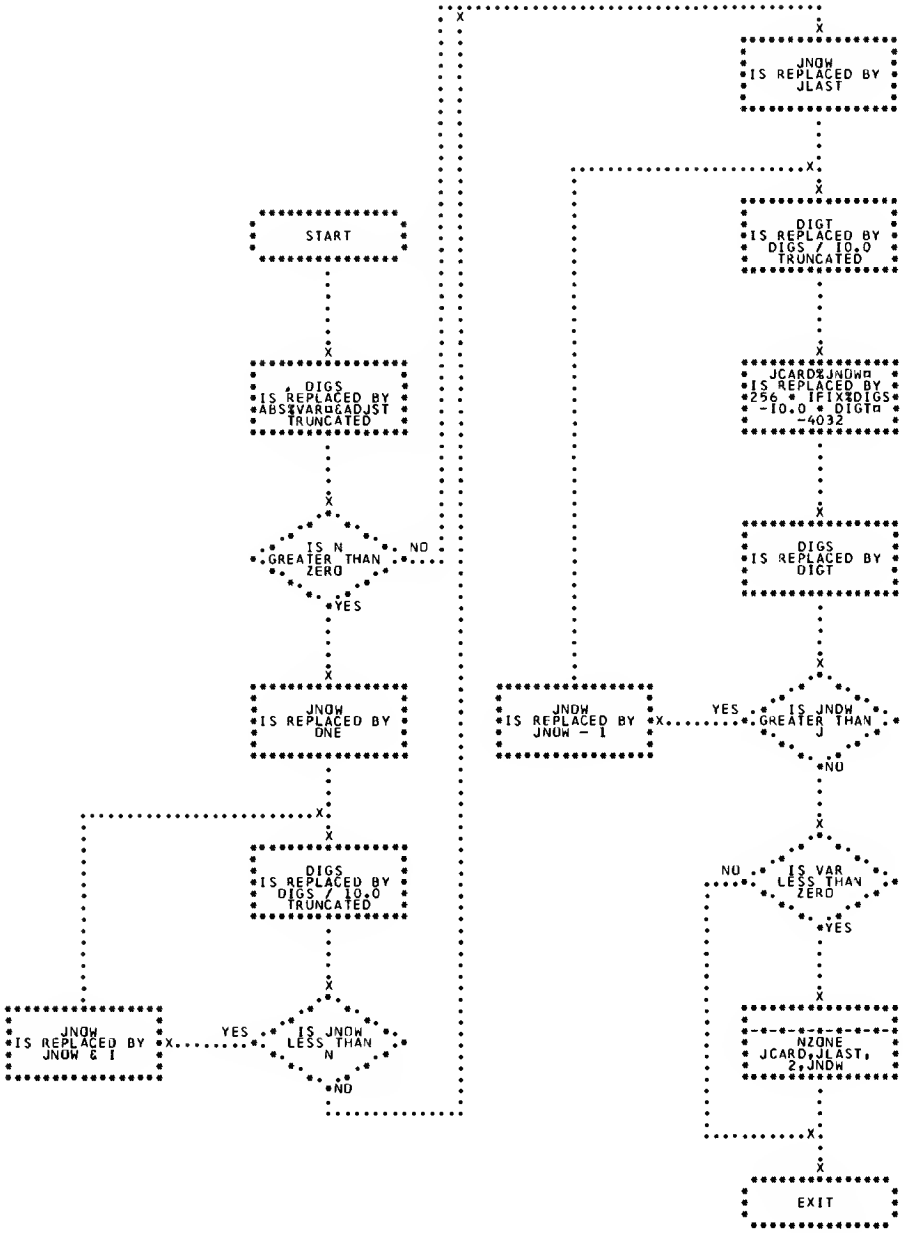


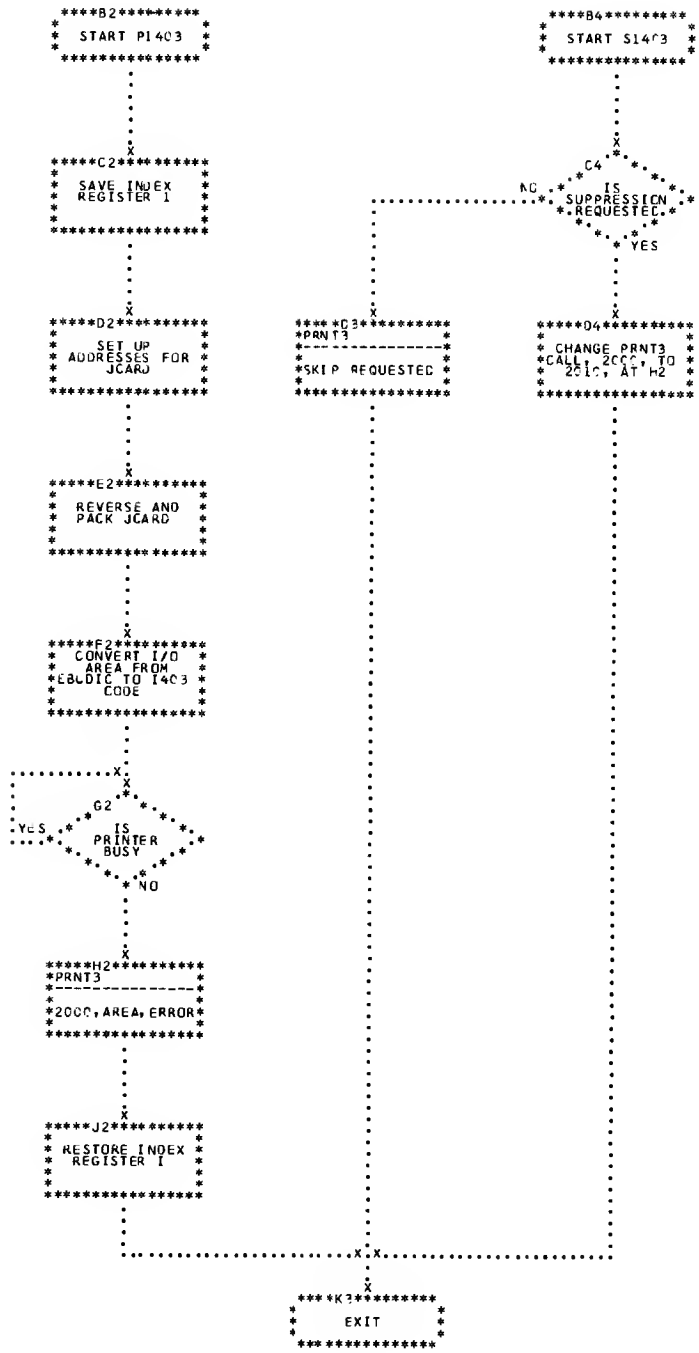
CHART NZ 1130 COMMERCIAL NZONE SUBROUTINE











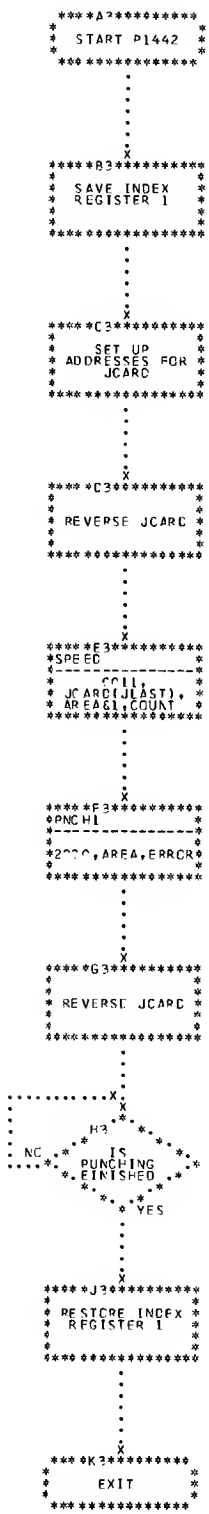
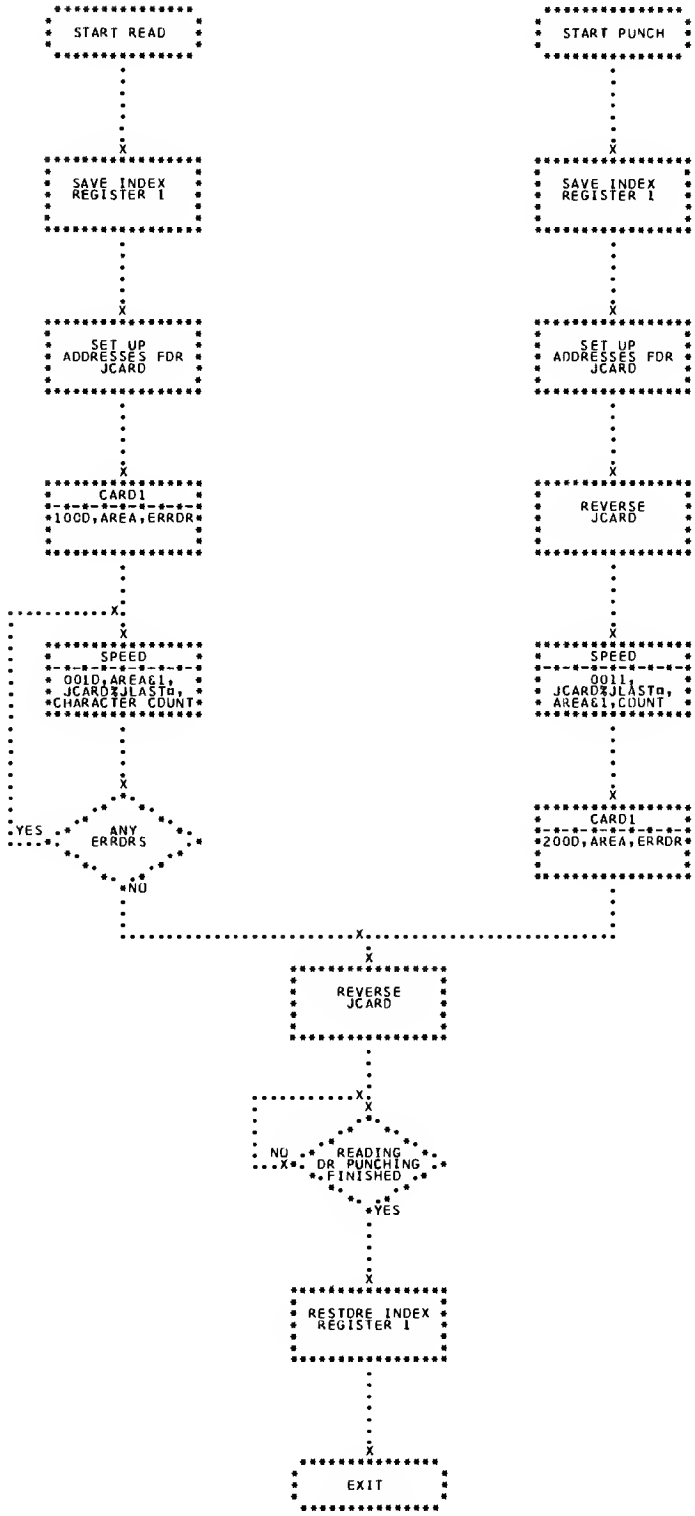
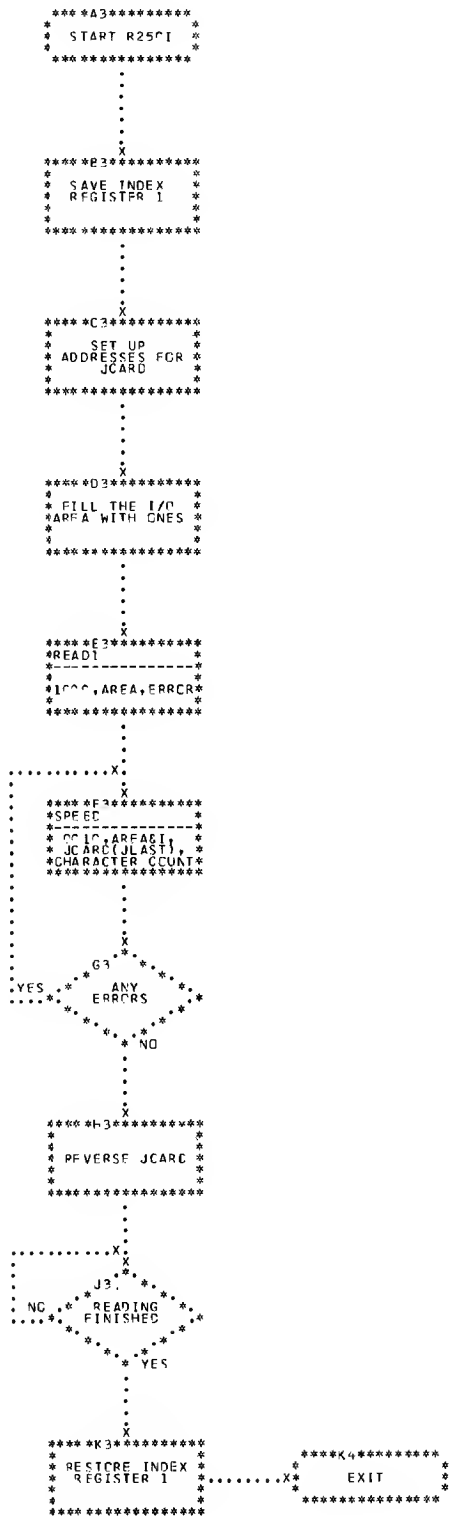


CHART RP 1130 COMMERCIAL READ/PUNCH SUBROUTINE




A3
START R2501
B3
SAVE INDEX REGISTER 1
C3
SET UP ADDRESSES FOR JCARD
D3
FILL THE I/O AREA WITH ONES
E3
READ
,AREA,ERROR
E3
SPEED
,AREA(1),
JCARD(LAST),
CHARACTER COUNT
G3
ANY ERRORS
YES
NO
H3
REVERSE JCARD
J3
READING FINISHED
NO
YES
K3
RESTORE INDEX REGISTER 1
K4
EXIT

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

CHART ST 1130 COMMERCIAL STACK SUBROUTINE

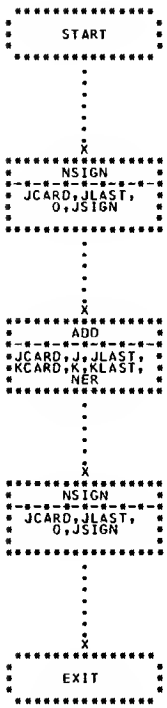
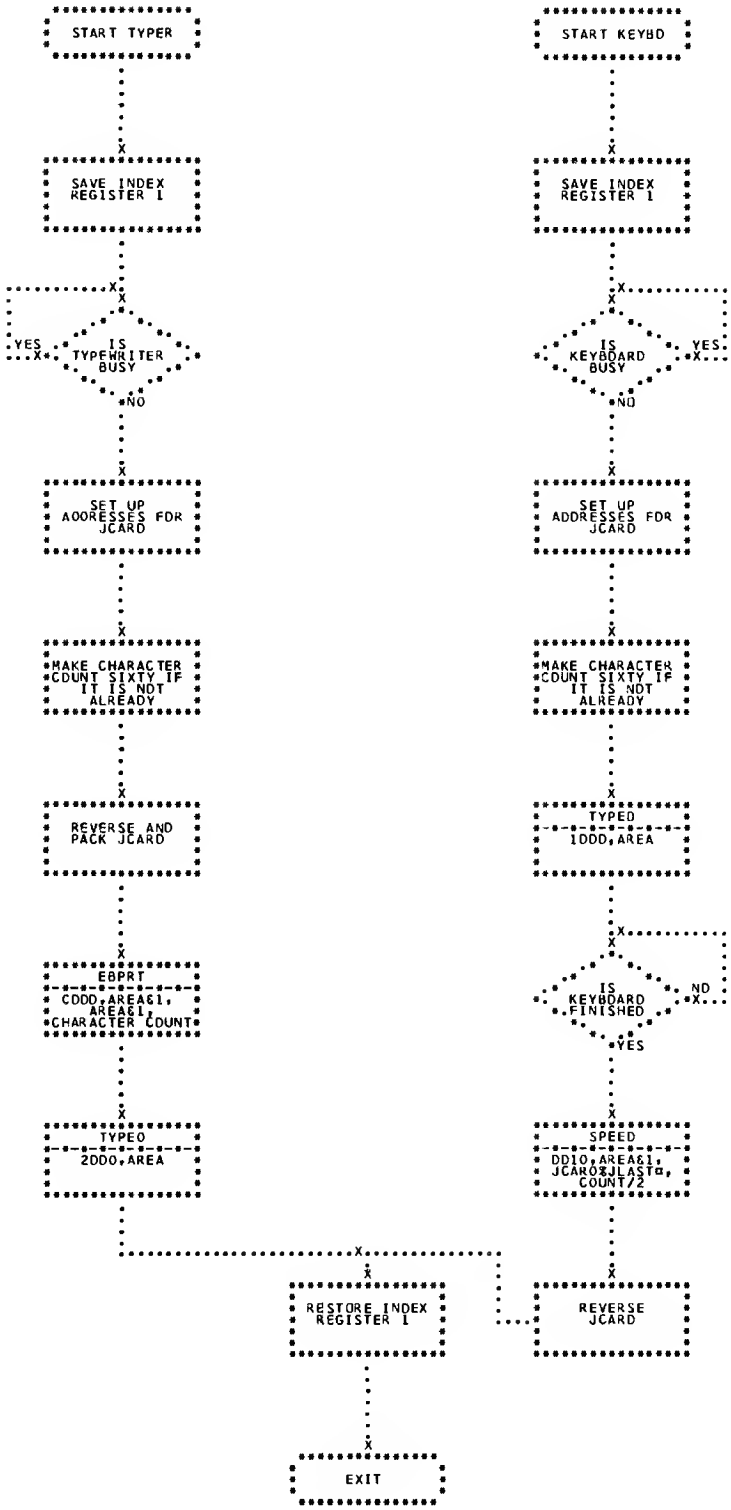
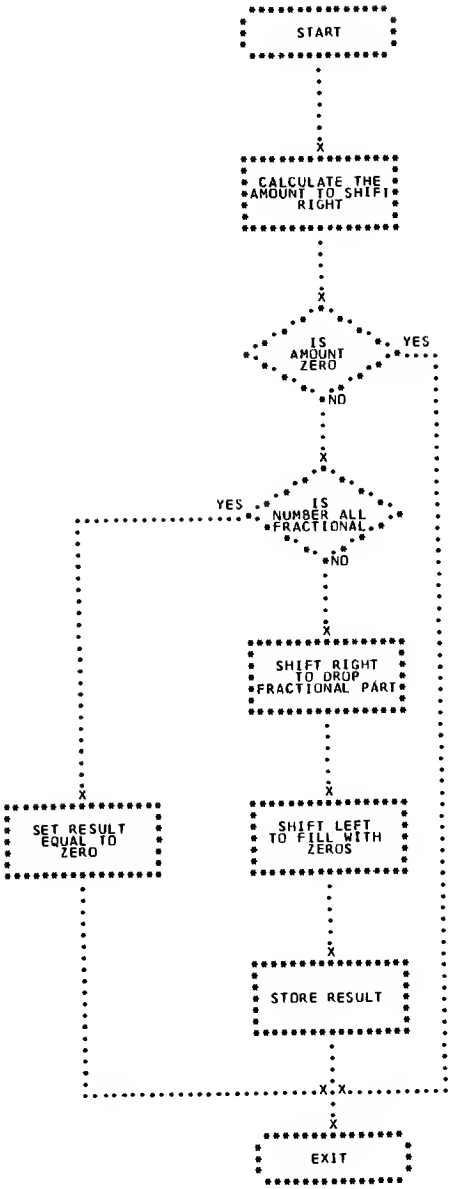


CHART TK 1130 COMMERCIAL TYPER/KEYBD SUBROUTINE





```
// JOB                                                                  CSP00010
// ASM                                                                  CSP00020
* NAME AOD                                                             (IO) CSP00030
** ADD/SUB SUBROUTINES FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (IO) CSP00040
* LIST                                                                 CSP00050
0008    01104000          ENT     AOO  AOD SUBROUTINE ENTRY POINT      CSP00060
                     *       CALL AOD(JCARD,J,JLAST,KCARD,K,KLAST,NER) CSP00070
                     *              THE FIELD JCARO(J) THROUGH        CSP00080
                     *              JCARO(JLAST) IS ADDED TO THE      CSP00090
                     *              FIELD KCARD(K) THROUGH            CSP00100
                     *              KCARD(KLAST).                     CSP00110
0000    22902000          ENT     SUB  SUBTRACT SUBROUTINE ENTRY POINT CSP00120
                     *       CALL SUB(JCARO,J,JLAST,KCARO,K,KLAST,NER) CSP00130
                     *              THE FIELO JCARD(J) THROUGH        CSP00140
                     *              JCARD(JLAST) IS SUBTRACTED FROM   CSP00150
                     *              THE FIELD KCARO(K) THROUGH        CSP00160
                     *              KCARD(KLAST).                     CSP00170
0000 0  0000        SUB   OC      *-*  ARGUMENT ADDRESS COMES IN HERE. CSP00180
0001 0  COFE              LD      SUB  PICK UP ARGUMENT AODRESS.       CSP00190
0002 0  D005              STO     AOO  STORE IT AT AOO.                CSP00200
0003 0  C002              LD      IHFS LOAD THE INSTRUCTION TO CHANGE  CSP00210
0004 0  0028              STO     SWIT SIGN OF JCARO FOR SUBTRACT.     CSP00220
0005 0  7005              MDX     ADO+3 START COMPUTING.               CSP00230
0006 0  F06E        IHFS  EOR  X  HFFFF-SWIT-1 CHANGE SIGN OF SUBTRHND CSP00240
0007 0  7002        MDX   MDX     *+2  SKIP OVER NEXT INSTRUCTION.     CSP00250
0008 0  0000        ADO   DC      *-*  ARGUMENT ADDRESS COMES IN HERE. CSP00260
0009 0  COFO              LD      MDX  LOAD SKIP OVER INSTRUCTION.     CSP00270
000A 0  D022              STO     SWIT STORE IT AT SWIT.               CSP00280
000B 0  6970              STX   1 SAVE1+1 SAVE IR1.                    CSP00290
000C 01 65800008          LDX   I1 ADD  PUT ARGUMENT ADDRESS IN IR1    CSP00300
000E 0  C100              LD    1 0    GET JCARO ADORESS               CSP00310
000F 00 95800002          S     I1 2    SUBTRACT JLAST VALUE           CSP00320
0011 0  D049              STO     DO+1 PLACE ADDRESS FOR ADD OR SUBTR  CSP00330
0012 0  8004              A       ONE+1 ADD CONSTANT OF ONE            CSP00340
0013 0  D017              STO     JPLUS+1 CREATE JCARD(JLAST) ADDRESS  CSP00350
0014 00 C5800002          LO    I1 2    GET JLAST VALUE                CSP00360
0016 00 95800001    ONE   S     I1 1    SUBTRACT J VALUE               CSP00370
0018 0  80FE              A       ONE+1 AOO CONSTANT OF ONE            CSP00380
0019 0  4B08              BSC     +    SKIP IF POSITIVE                CSP00390
001A 0  COFC              LD      ONE+1 NEGATIVE OR ZERO-MAKE COUNT 1  CSP00400
001B 0  003B              STO     COUNT+1 STORE JCARO LENGTH           CSP00410
001C 0  C103              LO    1 3    GET KCARO ADORESS               CSP00420
0010 0  D044              STO     KCRD1 PLACE IN CALLING SEQUENCE OF   CSP00430
001E 0  D062              STO     KCRD2 CARRY ANO FILL SUBROUTINES     CSP00440
001F 00 95800005          S     I1 5    SUBTRACT KLAST VALUE           CSP00450
0021 0  0037              STO     KCRD3+1 PLACE LOAD AOOR FOR AOO/SUB  CSP00460
0022 0  003A              STO     KCRD4+1 PLACE STORE ADOR FOR RESULT  CSP00470
0023 0  004F              STO     KCRD5+1 PLACE SUBTRACT AOORESS ANO   CSP00480
0024 0  0050              STO     KCR06+1 STORE ADDR FOR NEG CARRY     CSP00490
0025 0  B0F1              A       ONE+1 AOO CONSTANT OF ONE            CSP00500
0026 0  D044              STO     KCRD7+1 PLACE ADDR FOR SIGN CHANGE   CSP00510
0027 0  0010              STO     KPLUS+1 PLACE ADDR OF SIGN OF KCARO  CSP00520
0028 0  C106              LO    1 6    GET NER AODRESS                 CSP00530
0029 0  D05E              STO     ERA+1 SAVE NER AODRESS               CSP00540
                     *              CLEAR AND SAVE SIGNS ON JCARD     CSP00550
                     *              ANO KCARO FIELOS.                 CSP00560
002A 00 C4000000    JPLUS LD    L  *-*  GET SIGN OF JCARD               CSP00570
```

```
002C 0  D070            STO     JSIGN SAVE SIGN OF JCARD                CSP00580
002D 0  7002      SWIT  MDX     *+2  SKIP ON ADD=CHANGE SIGN ON SUBT     CSP00590
002E 01 D4800028        STO  I  JPLUS+1 STORE CHANGED SIGN OF JCARD     CSP00600
0030 01 4C100037        BSC  L  KPLUS,- DETERMINE SIGN OF JCARD         CSP00610
0032 0  F069            EOR     HFFFF NEGATIVE - MAKE POSITIVE          CSP00620
0033 01 D4800028        STO  I  JPLUS+1 STORE IT POSITIVE               CSP00630
0035 01 74010041        MDX  L  OP,1 CHANGE OPERATION - SEE OP & OPR    CSP00640
0037 00 C4000000  KPLUS LO   L  *-*  GET SIGN OF KCARD                  CSP00650
0039 0  D064            STO     KSIGN SAVE SIGN OF KCARD                CSP00660
003A 01 4C100041        BSC  L  OP,- DETERMINE SIGN OF KCARD            CSP00670
003C 0  F05F            EOR     HFFFF NEGATIVE - MAKE POSITIVE          CSP00680
003D 01 D4800038        STO  I  KPLUS+1 STORE IT POSITIVE               CSP00690
003F 01 74010041        MDX  L  OP,1 CHANGE OPERATION - SEE OP & OPR    CSP00700
                  *                     CALCULATE THE OPERATION.        CSP00710
                  *                     INITIALLY THIS IS FOR ADD.  IT  CSP00720
                  *                     CAN BE CHANGED UP TO TWO TIMES, CSP00730
                  *                     FIRST TO SUBTRACT AND THEN BACK CSP00740
                  *                     AGAIN TO ADD.  SEE OPR.         CSP00750
0041 0  C062      OP    LD        OPR  PICK UP OPERATION                  CSP00760
0042 0  D017            STO       DO   STORE IT AT DO                     CSP00770
0043 0  C063            LD        OPO  RESET THE PICK UP INSTRCTN TO +    CSP00780
0044 0  D0FC            STO       OP   WITH INSTRUCTION AT OPO            CSP00790
0045 0  C104            LD     1  4    GET ADDRESS OF K                  CSP00800
0046 0  D01C            STO       K1   STORE IT AT K1 FOR CARRY SUBRTN    CSP00810
0047 0  D03A            STO       K2   AND AT K2 FOR FILL SUBROUTINE      CSP00820
                  *                     DETERMINE IF JCARD IS LONGER    CSP00830
                  *                     THAN KCARD.  KLAST=JLAST+J=KNOW CSP00840
                  *                     IS COMPARED TO K.  IF KNOW IS   CSP00850
                  *                     GREATER THAN OR EQUAL TO K GO   CSP00860
                  *                     TO KLAS3 FOR ERROR.             CSP00870
0048 00 C5800005        LD     I1 5     GET KLAST VALUE                 CSP00880
004A 0  D03B            STO       KLAS3+1 SAVE IT TO INDICATE ERROR     CSP00890
0048 00 95800004        S      I1 4     SUBTRACT K VALUE                CSP00900
004D 0  D021            STO       COMP+1 SAVE FOR CMPLMNT ON NEG CARRY  CSP00910
004E 00 95800002        S      I1 2     SUBTRACT JLAST VALUE            CSP00920
0050 00 85800001        A      I1 1     ADD J VALUE                     CSP00930
0052 01 4C2800A0        BSC    L  RETAD,+2 IS JCARD LONGER THAN KCARD    CSP00940
0054 0  7107            MDX     1 7     NO=OK=MOVE OVER SEVEN ARGUMENTS CSP00950
0055 0  6928            STX     1 DONE1+1 CREATE RETURN ADDRESS         CSP00960
                  *                     SETUP JNOW                      CSP00970
0056 00 65000000  COUNT LDX    L1 *-*   LOAD JCARD LENGTH TO IR1         CSP00980
                  *                     KCARD(KNOW)=KCARD(KNOW) + OR -  CSP00990
                  *                                JCARD(JNOW)          CSP01000
0058 00 C5000000  KCRD3 LD     L1 *-*   LOAD KCARD(KNOW)                 CSP01010
005A 00 85000000  DO    A      L1 *-*   ADD OR SUBTRACT JCARD(JNOW)      CSP01020
005C 00 D5000000  KCRD4 STO    L1 *-*   STORE RESULT IN KCARD(KNOW)      CSP01030
                  *                     KNOW=KNOW+1 AND SEE IF JNOW IS  CSP01040
                  *                     GREATER THAN JLAST.  IF NOT,    CSP01050
                  *                     JNOW=JNOW+1 AND GO BACK FOR     CSP01060
                  *                     MORE.                           CSP01070
005E 0  71FF            MDX     1 -1    DECREMENT IR1                   CSP01080
005F 0  70F8            MDX       KCRD3 GO BACK FOR MORE                CSP01090
                  *                     RESOLVE CARRIES GENERATED       CSP01100
                  *                     DURING OPERATION.               CSP01110
0060 30 03059668  AGAIN CALL      CARRY GO TO CARRY SUBROUTINE            CSP01120
```

```
0062 0  0000       KCRD1 DC      *-*  KCARD ADDRESS                          CSP01130
0063 0  0000       K1    DC      *-*  K ADDRESS                              CSP01140
0064 I  0087       KLAS1 DC      KLAS3+1 KLAST ADDRESS                       CSP01150
0065 1  0008             DC      A00  ADDRESS TO HOLD ANY CARRY             CSP01160
                   *                   LET KNOW BE ANY RESULTING CARRY      CSP01170
                   *                   IF NEGATIVE, COMPLIMENT AND          CSP01180
                   *                   CHANGE THE SIGN OF KCARD.  IF        CSP01190
                   *                   ZERO, ALL DONE.  IF POSITIVE,        CSP01200
                   *                   OVERFLOW ERROR.                      CSP01210
0066 01 4C18008A          BSC  L  FIN,+- CHECK FOR ZERO-YES GO TO FIN         CSP01220
0068 01 4CI00080          BSC  L  ERR9,- NO-CHECK FOR OVERFLOW-YES ERR9       CSP01230
006A 00 84000000   KCRD7  A     L  *-*  COMPLIMENT-ADD CARRY TO LOW          CSP01240
006C 01 D480006B          STO  I  KCRD7+1 ORDER AND STORE IT BACK            CSP01250
                   *                   COMPLIMENT - SUBTRACT EACH           CSP01260
                   *                   DIGIT FROM 9 AND CHANGE THE          CSP01270
                   *                   SIGN OF KCARD.                       CSP01280
006E 00 65000000   COMP   LDX  L1 *-*  LOAD IR1 WITH LENGTH OF KCARD         CSP01290
0070 0  7I01             MDX   1 I    ADD 1 TO GET THE TRUE LENGTH           CSP01300
0071 0  C02E             LD        NINE LOAD A NINE.                        CSP01310
0072 00 95000000   KCRD5  S     LI *-*  SUBTRACT KCARD(KNOW)                 CSP01320
0074 00 05000000   KCRD6  STO   LI *-*  PUT BACK IN KCARD(KNOW)              CSP01330
                   *                   SEE IF KNOW IS GREATER THAN          CSP01340
                   *                   KLAST.  IF NOT, KNOW=KNOW+1          CSP01350
0076 0  71FF             MDX   1 -1   DECREMENT IR1                         CSP01360
0077 0  7CF9             MDX      COMP+3 GO BACK FOR MORE                   CSP01370
0078 0  C026             LD       KSIGN                                     CSP01380
0079 0  F0FA             EOR      KCRD6                                     CSP01390
007A 0  D024             STO      KSIGN SET SIGN OF KCARD                   CSP01400
007B 0  72E4             MDX      AGAIN CHECK AGAIN FOR CARRIES             CSP01410
007C 00 65000000   SAVE1  LDX  L1 *-*  RESTORE IR1                          CSP01420
007E 30 4C000000   DONE1  BSC  L  *-*  RETURN TO CALLING PROGRAM             CSP01430
                   *                   ERROR - ERROR - OVERFLOW- - - -      CSP01440
0080 30 062534C0   ERR9   CALL     FILL FILL KCARD WITH NINES.               CSP01450
0082 0  0000       KCRD2  DC       *-*  ADDRESS OF KCARD                     CSP01460
0083 0  0000       K2     DC       *-*  ADDRESS OF K                         CSP01470
0084 I  0087       KLAS2  OC       KLAS3+1 ADDRESS KLAST                     CSP01480
0085 I  00A0             DC       NINE FILL CHARACTER                       CSP01490
0086 00 65000000   KLAS3  LDX  L1 *-*  PICK UP KLAST VALUE                   CSP01500
0088 00 60000000   ERA    STX  L1 *-*  STORE VALUE AT NER                    CSP01510
                   *                   RESTORE SIGNS ON JCARD AND           CSP01520
                   *                   KCARD FIELDS                         CSP01530
008A 0  C0I3       FIN    LD       JSIGN PICK UP SIGN OF JCARD               CSP01540
008B 01 D480002B          STO  I   JPLUS+1 AND RESTORE IT                    CSP01550
008D 0  C011              LD       KSIGN PICK UP SIGN OF KCARD               CSP01560
008E 01 4C280095          BSC  L   NEG,+Z CHECK FOR PLUS OR MINUS            CSP01570
0090 01 C4800038          LD   I   KPLUS+1 PLUS-GET NEW SIGN AND             CSP01580
0092 01 4C280099          BSC  L   REV,+Z REVERSE IT IF NEGATIVE             CSP01590
0094 0  7DE7              MDX      SAVE1 POSITIVE-ALL OONE-GO TO EXIT..      CSP01600
0095 01 C4800038   NEG    LO   I   KPLUS+1 MINUS-GET NEW SIGN ANO            CSP01610
0097 01 4C28007C          BSC  L   SAVE1,+Z GO TO EXIT IF NOT NEGATIVE       CSP01620
0099 0  F003       REV    EOR      HFFFF REVERSE THE SIGN                    CSP01630
009A 01 D4800038          STO  I   KPLUS+1 STORE IT BACK                     CSP01640
009C 0  70DF              MDX      SAVE1 ALL DONE-GO TO EXIT.............    CSP01650
009D 0  FFFF       HFFFF  DC       /FFFF CONSTANT OF ALL BINARY ONES         CSP01660
009E 0  000D       JSIGN  OC       *-*  SIGN OF JCARD                        CSP01670
```



```
009F 0  000D       KSIGN  DC       *-*  SIGN OF KCARD                        CSP0168D
00A0 0  00D9       NINE   OC       9    CONSTANT OF NINE                    CSP01690
00A1 D  7107       RETAO  MOX    1 7          MOVE OVER SEVEN ARGUMENTS     CSP01700
00A2 0  69DC              STX    1 DONE1+1 CREATE RETURN ADDRESS             CSP01710
00A3 01 4C000086          BSC  L   KLAS3   GO TO KLAS3                      CSP01720
00A5 00 85D0000D   OPR    A    L1 *-*  ADD FOR ADD OR SUBTRACT OPERATN       CSP01730
00A7                      ORG      OPR+1 RESET THE ADDRESS COUNTER           CSP01740
D0A6 0D 95000000          S    L1 *-*  SUBTR FOR ADD OR SUBTR OPRATN         CSP01750
00A8                      ORG      OPR+2 RESET THE ADDRESS COUNTER           CSP01760
00A7 00 85000000          A    L1 *-*  AOD FOR ADD OR SUBTRACT OPERATN       CSP01770
D0A9                      ORG      OPR+3 RESET THE ADDRESS COUNTER           CSP01780
00A8 0  C063       OPO    LO   X   OPR-OP-1 FOR RESETING THE INSTRCTN       CSP01790
                   *                      AT OP TO ITS INITIAL STATE..      CSP01800
00AA                      END                                                CSP01810
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                     CSP01820
*STORE      WS  UA  ADD                                                    CSP01830
3418 000C
```

```
// ASM                                                                  CSP01840
** A1A3/A3A1 SUBROUTINES FOR 1130 COMMERCIAL SUBROUTINE PACKAGE    (10) CSP01850
* NAME A1A3                                                          (10) CSP01860
* LIST                                                                   CSP01870
0000     01C41CC0          ENT     A1A3 A1A3 SUBROUTINE ENTRY POINT     CSP01880
                     *     CALL A1A3(JCARO,J,JLAST,KCARD,K,ICHAR)       CSP01890
                     *          THE WORDS JCARD(J) THROUGH              CSP01900
                     *          JCARD(JLAST) IN A1 FORMAT ARE           CSP01910
                     *          CRAMMED INTO KCARD IN A3 FORMAT.        CSP01920
0006     01CC1C40          ENT     A3A1 A3A1 SUBROUTINE ENTRY POINT     CSP01930
                     *     CALL A3A1(JCARD,J,JLAST,KCARO,K,ICHAR)       CSP01940
                     *          THE WORDS JCARO(J) THROUGH              CSP01950
                     *          JCARO(JLAST) IN A3 FORMAT ARE           CSP01960
                     *          UNCRAMMED INTO KCARD IN A1 FORMAT.      CSP01970
0000 0   0000       A1A3  DC        *-* ARGUMENT ADDRESS COMES IN HERE   CSP01980
0001 0   C002             LD      SW1 LOAD BRANCH TO ELSE                CSP01990
0002 0   002A             STO     SWTCH STORE BRANCH AT SWITCH           CSP02000
0003 0   7007             MDX     START  START COMPUTING                 CSP02010
0004 0   7021       SW1   MDX  X  ELSE-SWTCH-1  BRANCH TO ELSE          CSP02020
0005 0   7000       SW2   MDX  X  0  NOP INSTRUCTION                     CSP02030
0006 0   0000       A3A1  DC        *-* ARGUMENT ADDRESS COMES IN HERE   CSP02040
0007 0   C0FE             LD      A3A1 PICK UP ARGUMENT ADDRESS AND      CSP02050
0008 0   D0F7             STO     A1A3 STORE IT IN A1A3                  CSP02060
0009 0   C0FB             LD      SW2  LOAD NOP INSTRUCTION              CSP02070
000A 0   0022             STO     SWTCH  STORE NOP AT SWITCH             CSP02080
000B 0   6965       START STX   1 SAVE1+1  SAVE IR1                      CSP02090
000C 0   6A66             STX   2 SAVE2+1  SAVE IR2                      CSP02100
000D 0   6B67             STX   3 SAVE3+1  SAVE IR3                      CSP02110
000E 01  65800000         LDX   I1 A1A3 PUT ARGUMENT ADDRESS IN IR1      CSP02120
0010 0   C100             LD    1 0  GET JCARD ADDRESS                   CSP02130
0011 00  95800002         S     I1 2  SUBTRACT JLAST VALUE               CSP02140
0013 0   0018             STO     JCARD+1  CREATE JCARD(J) ADDRESS       CSP02150
0014 0   D03F             STO     OVR1+1   STORE JCARD(J) ADDRESS        CSP02160
0015 0   0044             STO     OVR2+1   STORE JCARO(J) ADDRESS        CSP02170
0016 0   C103             LD    1 3  GET KCARD ADDRESS                   CSP02180
0017 0   8006             A       ONE+1  ADD CONSTANT OF 1               CSP02190
0018 00  95800004         S     I1 4  SUBTRACT K VALUE                   CSP02200
001A 0   D00D             STO     KCARO+1  CREATE KCARD(K) ADDRESS       CSP02210
001B 00  C5800002         LD    I1 2  GET JLAST VALUE                    CSP02220
001D 00  95800001   ONE   S     I1 1  SUBTRACT J VALUE                   CSP02230
001F 0   80FE             A       ONE+1  ADD CONSTANT OF 1               CSP02240
0020 0   D009             STO     CNT+1  CREATE FIELD WIDTH              CSP02250
0021 0   C105             LD    1 5  GET ICHAR ADDRESS                   CSP02260
0022 0   9028             S       D40    SUBTRACT CONSTANT OF 40         CSP02270
0023 0   D060             STO     TABLE+1  CREATE TABLE END ADDRESS      CSP02280
0024 0   0066             STO     TCODE+1  STORE TABLE END ADDRESS       CSP02290
0025 0   7106             MDX   1 6  ADJUST OVER 6 ARGUMENTS             CSP02300
0026 0   6950             STX   1 DONE1+1  CREATE RETURN ADDRESS         CSP02310
0027 00  65000000   KCARO LDX   L1 *-*  PUT KCARD ADDRESS IN IR1          CSP02320
0029 00  66000000   CNT   LDX   L2 *-*  PUT FIELD WIDTH IN IR2            CSP02330
002B 00  C6000000   JCARD LD    L2 *-*  PICK UP JCARO(J)                 CSP02340
002D 0   7000       SWTCH MDX  X  0  SWITCH BETWEEN CRAM AND UNCM        CSP02350
002E 01  4C280047         BSC   L MINUS,+Z  TEST SIGN OF INTEGER         CSP02360
0030 0   1890             SRT     16   SHIFT INTEGER TO EXTENSION       CSP02370
0031 0   A818             O       D1600 DIVIDE BY 1600                   CSP02380
0032 0   801B             A       020  ADJUST FIRST VALUE                CSP02390
0033 0   D0D2       HOLD  STO     A3A1 SAVE FIRST CHARACTER VALUE         CSP02400
```

```
0034 0  1810              SRA     16   ZERO ACCUMULATOR                    CSP02410
0035 0  A815              D       D40  DIVIDE BY 4D                        CSP02420
0036 0  D0C9              STO     A1A3 SAVE SECOND CHARACTER VALUE         CSP02430
0037 D  1090              SLT     16 SHIFT THIRD CHAR VALUE TO ACCUM       CSP02440
0038 01 4400007E          BSI  L  DECOD DECODE THIRD CHARACTER             CSP02450
003A D  D1FE              STO   1 -2   STORE THIRD CHARACTER                CSP02460
003B 0  C0C4              LD      A1A5 GET SECOND CHARACTER                CSP02470
003C 01 44D0007E          BSI  L  DECOD DECODE SECOND CHARACTER            CSP02480
003E D  D1FF              STO   1 -1   STORE SECOND CHARACTER              CSP02490
003F 0  C0C6              LD      A3A1 GET FIRST CHARACTER                 CSP02500
0040 01 4400007E          BSI  L  DECOD DECODE FIRST CHARACTER             CSP02510
0042 0  D100              STO   1 0    STORE FIRST CHARACTER               CSP02520
0043 0  71FD              MDX   1 -3   DECREMENT A1 OUT ARRAY              CSP02530
0044 0  72FF              MDX   2 -1   DECREMENT FIELD WIDTH               CSP02540
0045 0  70E5              MDX     JCARD FIELD WIDTH IS NOT 2ERO            CSP02550
0046 0  7029              MDX     SAVE1 GO TO RESTORE AND RETURN           CSP02560
0047 0  8004        MINUS A       D32K ADJUST FOR NEGATIVE INTEGER         CSP02570
0048 0  1890              SRT     16   SHIFT INTEGER TO EXTENSION          CSP02580
0049 0  A803              D       D1600 DIVIDE BY 1600                     CSP02590
004A 0  70E8              MDX     HOLD GO TO GET THE REMAINING INTEGERS    CSP02600
004B 0  0028        D40   DC      40   CONSTANT OF 40                      CSP02610
004C 0  7D00        D32K  DC      32000 CONSTANT OF 32000                  CSP02620
004D 0  0640        D1600 DC      1600 CONSTANT OF 1600                    CSP02630
004E 0  0014        D20   DC      20   CONSTANT OF 20                      CSP02640
004F 0  D0B6        ELSE  STO     A3A1 STORE FIRST A1 CHARACTER            CSP02650
0050 D  72FF              MDX   2 -1   DECREMENT FIELD WIDTH               CSP02660
0051 0  7001              MDX     OVR1 GO TO GET NEXT CHARACTER            CSP02670
0052 0  7025              MDX     FILL1 LAST CHARACTER-FILL WITH BLANK     CSP02680
0053 0D C6000000    OVR1  LD    L2 *-*  GET SECOND CHARACTER                CSP02690
0055 0  D0AA              STO     A1A3 STORE SECOND CHARACTER              CSP02700
0056 0  72FF              MDX   2 -1   DECREMENT FIELD WIDTH               CSP02710
0057 0  7001              MDX     OVR2 GO TO GET NEXT CHARACTER            CSP02720
0058 0  7021              MDX     FILL2 LAST CHARACTER-FILL BLANK          CSP02750
0059 00 C6000000    OVR2  LD    L2 *-*  GET THIRD CHARACTER                 CSP02740
005B 01 44000087    RET   BSI  L  CODE CODE CHARACTER TO NUMBER             CSP02750
005D 0  D0CA              STO     KCARD61 SAVE NUMBR OF THIRD CHARACTER    CSP02760
005E 0  C0A1              LD      A1A3 GET SECOND CHARACTER                CSP02770
005F 01 44000087          BSI  L  CODE CODE SECOND CHARACTER               CSP02780
0061 D  A0E9              M       D40  MULTIPLY BY 40 AND                  CSP02790
0062 0  1090              SLT     16   SHIFT TO ACCUMULATOR                CSP02800
0063 D  80C4              A       KCARD+1 ADD NUMBER(THIRD) AND            CSP02810
0064 0  D0C5              STO     KCARD+1 SAVE RESULTING INTEGER           CSP02820
0065 0  C0A0              LD      A3A1 GET FIRST CHARACTER                 CSP02830
0066 01 44000087          BSI  L  CODE CODE FIRST CHARACTER                CSP02840
0068 0  90E5              S       D20  SUBTRACT 20                         CSP02850
0069 0  ADE3              M       D1600 MULTIPLY BY 1600                   CSP02860
006A 0  1090              SLT     16   SHIFT TO ACCUMULATOR                CSP02870
006B 0  80BC              A       KCARD+1 ADD IN PREVIOUS RESULT           CSP02880
006C 0  D100              STO   1 0    STORE IN A3 ARRAY                   CSP02890
006D 0  71FF              MDX   1 -1   NEXT WORD IN A3 ARRAY               CSP02900
006E 0  T2FF              MDX   2 -1   DECREMENT FIELD WIDTH               CSP02910
006F 0  70B8              MDX     JCARD GET MORE A1 CHARACTERS             CSP02920
0070 00 65000000    SAVE1 LDX  L1 *-*  RESTORE IR1                         CSP02930
0072 00 66000000    SAVE2 LDX  L2 *-*  RESTORE IR2                         CSP02940
0074 00 67000000    SAVE5 LDX  L3 *-*  RESTORE IR3                         CSP02950
```



```
0076 00 4C000000    DONE1 BSC  L  *-*  RETURN TO CALLING PROGRAM           CSP02960
0078 0  C004        FILL1 LD      H4040 FILL WITH TWO BLANKS                CSP02970
007A 0  D086              STO     A1A5 STORE SECOND CHARACTER BLANK        CSP02980
007A 0  C002        FILL2 LD      H4040 FILL WITH ONE BLANK                 CSP02990
007B 0  7201              MDX   2 1    SET IR1 TO 1                       CSP03000
007C 0  70DE              MDX     RET  GO TO CODE ROUTINE                  CSP03010
007D 0  4040        H4040 DC      /4040 CONSTANT OF A1 BLANK               CSP03020
007E 0  0000        DECOD DC      *-*  DECODE RETURN ADDRESS GOES HERE     CSP03030
007F 0  809E              A       ONE+1 ADD ONE TO NUMBER GIVING           CSP03040
0080 0  D001              STO     PLACE+1 SUBSCRIPT OF TABLE AND SAVE      CSP03050
0081 00 67000000    PLACE LDX  L3 *-*  LOAD IR3 WITH SUBSCRIPT OF TABLE    CSP03060
0083 00 C7000000    TABLE LD   L3 *-*  GET A1 CHARACTER                     CSP03070
0085 01 4C80D07E          BSC  I  DECOD RETURN                             CSP05080
0087 0  0000        CODE  DC      *-*  CODE RETURN ADDRESS GOES HERE       CSP05090
0088 0  D0F5              STO     DECOD SAVE THE CHARACTER TO BE CODED     CSP05100
D089 0  6328              LDX   3 40 LOAD IR3 WITH THE TABLE LENGTH=40     CSP03110
008A 00 C7000000    TCODE LD   L3 *-*  LOAD CHARACTER FROM ICHAR ARRAY     CSP03120
008C 0  F0F1              EOR     DECOD ZERO ACCUMULATOR IF MATCH          CSP03150
008D 01 4C2D0094          BSC  L  OUT+Z GO TO PUT IF NOT 2ERO              CSP05140
008F 0  6BEE        AWAY  STX   3 DECOD SAVE SUBSCRIPT OF MATCH            CSP03150
0090 0  C0ED              LD      DECOD LOAD SUBSCRIPT                     CSP03160
0091 0  908C              S       ONE+1 SUBTRACT ONE GIVING NUMBER         CSP03170
0092 01 4C80008T          BSC  I  CODE RETURN                              CSP03180
0094 0  73FF        OUT   MDX   3 -1 DECREMENT THROUGH THE TABLE-ICHAR     CSP03190
0095 0  70F4              MDX     TCODE GO TRY AGAIN                       CSP03200
0096 0  C0E6              LD      H4040 NOT IN THE TABLE - LOAD A BLANK    CSP03210
0097 0  T0F0              MDX     CODE+1 GO BACK TO CODE THE BLANK....     CSP03220
0098                      ENO                                              CSP03230
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP
*STORE       WS  UA  A1A3                                                  CSP03240
3332 000A                                                                  CSP03250
```

```
// ASM                                                                  CSP03260
** A1DEC SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP03270
* NAME A1DEC                                                        (ID) CSP03280
* LIST                                                                 CSP03290
0001     01C44143          ENT     A1DEC A1DEC SUBROUTINE ENTRY POINT   CSP03300
                     *                    CALL A1DEC(JCARD,J,JLAST,NER)  CSP03310
                     *                    THE WORDS JCARD(J) THROUGH     CSP03320
                     *                    JCARD(JLAST) ARE CONVERTED FROM CSP03330
                     *                    A1 FORMAT TO 01 FORMAT AND THE CSP03340
                     *                    ORIGINAL DATA IS REPLACED BY THE CSP03350
                     *                    CONVERTED DATA.                CSP03360
0000 0   0004       FDUR    DC      4    CONSTANT OF FDUR               CSP03370
0D01 0   0000       A1DEC   DC      *-*  ARGUMENT ADDRESS COMES IN HERE CSP03380
0002 0   6941              STX   1 SAVE1+1 SAVE IR1                     CSP03390
0003 01  65800001          LDX   I1 A1DEC PUT ARGUMENT ADDRESS IN IR1   CSP03400
0005 0   C100              LD    1 0    GET JCARD ADDRESS               CSP03410
0006 0   D017              STO     JCRD1 SETUP JCARD ADDRESS FOR NZONE  CSP03420
0007 00  9580D002  TWD     S     I1 2    SUBTRACT JLAST VALUE           CSP03430
0009 0   D01B              STO     PICK+1 PLACE LDAD ADDRESS FOR CONVRS CSP03440
000A 0   D02C              STO     PUT+1 PLACE STORE ADDRESS FOR CONVRS CSP03450
000B 0   8007              A       ONE+1 ADD CONSTANT OF ONE            CSP03460
000C 0   D033              STO     LAST+1 PLACE ADDRESS OF SIGN POSITON CSP03470
000D 0   C1D2              LD    1 2    GET JLAST ADDRESS              CSP03480
000E 0   D010              STD     JLAS1 SETUP JLAST ADDRESS FOR NZONE  CSP03490
000F 01  C480001F          LD    I JLAS1 GET JLAST VALUE AND           CSP03500
0011 0   D0EF              STO     A1DEC SAVE IT AT A1DEC               CSP03510
0012 00  95800001  ONE     S     I1 1    SUBTRACT J VALUE               CSP03520
0014 0   80FE              A       ONE+1 ADD CONSTANT OF ONE            CSP03530
0015 0   4808              BSC     +    CHECK FIELD WIDTH               CSP03540
0016 0   C0FC              LD      ONE+1 ZERO OR NEGATIVE=MAKE IT ONE   CSP03550
0017 0   D00B              STD     CDUNT+1 OK-SAVE WIDTH IN COUNT       CSP03560
0018 0   C103              LD    1 3    GET NER ADDRESS                 CSP03570
0019 0   D016              STD     ERA+1 SAVE IT                        CSP03580
001A 0   7104              MDX   1 4    MOVE OVER FOUR ARGUMENTS        CSP03590
001B 0   692A              STX   1 DONE1+1 CREATE RETURN ADDRESS        CSP03600
                     *                    REMOVE AND SAVE THE SIGN       CSP03610
001C 30  15A56545          CALL    NZONE REMDVE THE ZONE OVER LOW ORDER CSP03620
001E 0   0000      JCRD1   DC      *-*  ADDRESS OF JCARD                CSP03630
001F 0   0000      JLAS1   DC      *-*  ADDRESS OF JLAST                CSP03640
0020 1   0000              DC      FDUR ADDRESS OF CONSTANT OF FOUR     CSP03650
0021 1   001E              DC      JCRD1 ADDRESS OF DLD ZONE            CSP03660
                     *                    JNOW=J                         CSP03670
0022 00  6500000D  COUNT   LDX   L1 *-*  LOAD IR1 WITH FIELD WIDTH      CSP03680
                     *                    JTEST=JCARD(JNOW)              CSP03690
0024 00  C5000000  PICK    LD    L1 *-*  PICK UP JCARD(JNDW) AND        CSP03700
0026 01  4C100032          BSC   L  PDS,- CHECK IT AGAINST ZERO         CSP03710
0028 0   901E              S       ZERD NEGATIVE=IS IT LESS THAN        CSP03720
0029 01  4C100035          BSC   L  OK,- AN EBCDIC ZERO                 CSP03730
                     *                    NER=JNOW                       CSP03740
002B 0   69F7      ERA     STX     1 COUNT+1 YES = ERROR                CSP03750
002C 0   C0D4              LD      A1DEC COMPUTE THE SUBSCRIPT          CSP03760
002D 0   90F5              S       COUNT+1 OF THIS CHARACTER IN         CSP03770
002E 0   80E4              A       DNE+1 THE ARRAY AND                  CSP03780
002F 00  D4000000  ERA     STD   L  *-*  STORE THE SUBSCRIPT AT NER     CSP03790
0031 0   7006              MDX     MORE GO GET THE NEXT CHARACTER       CSP03800
0D32 0   9015      POS     S       BLANK NOT NEGATIVE - IS IT AN        CSP03810
0033 01  4C20002B          BSC   L  ERR,Z EBCDIC BLANK                  CSP03820
```

```
                                                                  PAGE   2
                     *                    JTEST + 4032 IS NOW IN ACCUM   CSP03830
                     *                    SHIFT 8 IS SAME AS DIVIDE BY 256 CSP03840
0035 0   18D8      OK      SRA     8    EITHER BLANK OR DIGIT - PUT     CSP03850
0036 00  D5000000  PUT     STO   L1 *-*  THE FOUR BITS OF DECIMAL BACK  CSP03860
                     *                    SEE IF JNOW IS LESS THAN JLAST. CSP03870
                     *                    IF YES, JNOW=JNOW+1 AND GO BACK CSP03880
                     *                    FOR MORE.  IF NO, SET UP THE   CSP03890
                     *                    SIGN.                          CSP03900
0038 0   71FF      MORE    MDX     1 -1   DECREMENT THE FIELD WIDTH      CSP03910
0039 0   70EA              MDX     PICK GO BACK FOR MORE                CSP03920
                     *                    WAS THE ORIGINAL SIGN INDICATION CSP03930
                     *                    TWO.  IF NOT, ALL DONE.  IF YES CSP03940
                     *                    MAKE THE SIGN NEGATIVE.        CSP03950
                     *                    JCARD(JLAST)=-JCARD(JLAST) - 1 CSP03960
0D3A 0   C0E3              LD      JCRD1 PICK UP THE DLD ZONE AND       CSP03970
003B 0   90CC              S       TWO+1 CHECK IT AGAINST TWD           CSP03980
003C 01  4C200043          BSC   L  SAVE1,Z IF NO MATCH GO TO EXIT      CSP03990
003E 0   90D4              S       ONE+1 IF MATCH, MAKE THE             CSP04000
003F 00  F4000000  LAST    EDR   L  *-*  SIGN NEGATIVE(LOW ORDER) AND  CSP04010
0041 01  D4800040          STO   I  LAST+1 STORE IT BACK                CSP04020
                     *                    EXIT.........................  CSP04030
0043 00  6500D000  SAVE1   LDX   L1 *-*  RESTORE IR1                    CSP04040
0045 00  4C000000  DONE1   BSC   L  *-*  RETURN TO CALLING PROGRAM      CSP0405D
0047 0   F040      ZERO    OC      /F040 CONSTANT OF EBCDIC ZERD        CSP04060
0D48 0   4040      BLANK   OC      /4040 CONSTANT OF EBCDIC BLANK       CSP04070
004A               END                                                  CSP04080
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP04090
*STORE      WS  UA  A1DEC                                               CSP04100
333C 0D05
```

```
// ASM                                                                  CSP04110
** CARRY SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP04120
* NAME CARRY                                                         (IO) CSP04130
* LIST                                                                 CSP04140
0000    03059668            ENT     CARRY CARRY SUBROUTINE ENTRY POINT   CSP04150
                      *                 CALL CARRY(JCARD,J,JLAST,KARRY)  CSP04160
                      *                 THE WORDS JCARD(J) THROUGH       CSP04170
                      *                 JCARD(JLAST) ARE CHECKED TO SEE  CSP04180
                      *                 THAT THEY ARE BETWEEN ZERO AND   CSP04190
                      *                 NINE.  IF THEY ARE NOT, THE      CSP04200
                      *                 UNITS DIGIT REMAINS AND THE TENS CSP04210
                      *                 DIGIT IS TREATED AS A CARRY TO   CSP04220
                      *                 THE NEXT WORD.                   CSP04230
0000 0  0000        CARRY DC      *-* ARGUMENT ADDRESS COMES IN HERE     CSP04240
0001 0  6930              STX   1 SAVE1+1 SAVE IR1                       CSP04250
0002 01 65800000          LDX   I1 CARRY PUT ARGUMENT ADDRESS IN IR1     CSP04260
0004 0  C100              LD    1 0     GET JCARD ADDRESS                CSP04270
0005 00 95800002          S     I1 2    SUBTRACT JLAST VALUE             CSP04280
0007 0  8004              A       ONE+1 ADD CONSTANT OF ONE              CSP04290
0008 0  0011              STO     SRCE+1 CREATE JCARD(JLAST) ADDRESS     CSP04300
0009 00 C5800002          LD    I1 2    GET JLAST VALUE                  CSP04310
0008 00 95800001    ONE   S     I1 1    SUBTRACT J VALUE                 CSP04320
000D 0  80FE              A       ONE+1 ADD CONSTANT OF ONE              CSP04330
000E 0  4808              BSC     +     CHECK FIELD WIDTH                CSP04340
000F 0  C0FC              LD      ONE+1 ZERO OR NEGATIVE-MAKE IT ONE     CSP04350
0010 0  0007              STO     COUNT+1 OK-SAVE WIDTH IN COUNT         CSP04360
0011 0  C103              LD    1 3     GET KARRY ADDRESS                CSP04370
0012 0  D01D              STO     OVF+1 AND SAVE IT                      CSP04380
0013 0  7104              MDX   1 4     MOVE OVER FOUR ARGUMENTS         CSP04390
0014 0  691F              STX   1 DONE1+1 CREATE RETURN ADDRESS          CSP04400
0015 0  10A0              SLT     32    CLEAR THE ACCUMULATOR AND EXTEN  CSP04410
                      *                 LET CARRY BE THE SAME AS NCARY   CSP04420
0016 0  D0E9              STO     CARRY SET NCARY TO ZERO                CSP04430
0017 00 65000000    COUNT LDX   L1 *-*  LOAD IR1 WITH THE FIELD WIDTH     CSP04440
                      *                 THE NEXT INSTRUCTION STARTS OUT  CSP04450
                      *                 BY PICKING UP JCARD(JLAST).      CSP04460
                      *                 THE SUBSCRIPT IS DECREMENTED BY  CSP04470
                      *                 THE INSTRUCTION AFTER POSZ.      CSP04480
                      *                 THE CALCULATIONS ARE..           CSP04490
                      *                 JTEST=JCARD(JNOW)+NCARY          CSP04500
                      *                 NCARY=JTEST/10                   CSP04510
                      *                 JTEST=JTEST-10*NCARY             CSP04520
0019 00 C4000000    SRCE  LD    L *-*  PICK UP JCARD(JNOW)               CSP04530
0018 0  80E4              A       CARRY ADD THE PREVIOUS CARRY TO IT     CSP04540
001C 0  1890              SRT     16    SHIFT THE ACCUM TO THE EXTENTON  CSP04550
001D 0  A817              D       TEN   DIVIDE BY TEN AND                CSP04560
001E 0  D0E1              STO     CARRY STORE THE QUOTIENT AT NCARY      CSP04570
                      *                 THE QUOTIENT IS THE GENERATED    CSP04580
                      *                 CARRY.                           CSP04590
001F 0  1090              SLT     16    PUT REMAINDER IN ACCUMULATOR AN  CSP04600
0020 01 4C100028          BSC   L POSZ,- CHECK TO SEE IF NEGATIVE-NO-     CSP04610
                      *                 GO TO POSZ.....                  CSP04620
0022 0  8012              A       TEN   YES - COMPLIMENT BY ADDING TEN   CSP04630
0023 0  1890              SRT     16    STORE TEMPORARILY IN EXTENTION   CSP04640
0024 0  C008              LO      CARRY LOAD NCARY                       CSP04650
0025 0  90E6              S       ONE+1 AND SUBTRACT                     CSP04660
0026 0  D0D9              STO     CARRY ONE FROM IT                      CSP04670
```



```
                      *                 JCARD(JNOW)=JTEST                CSP04680
0027 0  1090              SLT     16    SHIFT COMPLIMENTED REMAINDER     CSP04690
                      *                 BACK TO ACCUMULATOR              CSP04700
0028 01 0480001A    POSZ  STO   I SRCE+1 AND STORE IN RESULT             CSP04710
                      *                 JNOW=JNOW-1                      CSP04720
002A 01 7401001A          MDX   L SRCE+1,1 GO TO NEXT DIGIT OF JCARD     CSP04730
                      *                 IF JNOW IS LESS THAN J, ALL      CSP04740
                      *                 DONE.  OTHERWISE, GET THE NEXT   CSP04750
                      *                 DIGIT.                           CSP04760
002C 0  71FF              MDX     1 -1  DECREMENT THE FIELD WIDTH        CSP04770
002D 0  70E8              MDX     SRCE GO BACK FOR NEXT DIGIT            CSP04780
                      *                 KARRY=NCARY                      CSP04790
002E 0  C0D1              LO      CARRY ALL DONE - PICK UP ANY           CSP04800
002F 00 04000000    OVF   STO   L *-*  GENERATED CARRY AND STORE IT      CSP04810
                      *                 AR KARRY.  EXIT.................. CSP04820
0031 00 65000000    SAVE1 LDX   L1 *-*  RESTORE IR1                      CSP04830
0033 00 4C000000    DONE1 BSC   L *-*  RETURN TO CALLING PROGRAM          CSP04840
0035 0  000A        TEN   DC      10    CONSTANT OF TEN                  CSP04850
0036                      END                                            CSP04860
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP04870
*STORE      WS  UA  CARRY                                              CSP04880
3341 0004
```

```
// ASM                                                                  CSP04890
** DECA1 SUBROUTINE FOR 113D COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP04900
* NAME DECA1                                                        (ID) CSP04910
* LIST                                                                 CSPD4920
0000    04143071          ENT     DECA1 DECA1 SUBROUTINE ENTRY POINT    CSP0493D
                     *                 CALL DECA1(JCARD,J,JLAST,NER)     CSP0494D
                     *                 THE WORDS JCARD(J) THROUGH        CSP0495D
                     *                 JCARD(JLAST) ARE CONVERTED FROM   CSP0496D
                     *                 D1 FORMAT TO A1 FORMAT AND THE    CSP04970
                     *                 ORIGINAL DATA IS REPLACED BY THE  CSP04980
                     *                 CONVERTED DATA.                   CSP04990
0000 0  0000        DECA1 DC      *-*   ARGUMENT ADDRESS COMES IN HERE    CSP0500D
0001 0  6942              STX     1 SAVE1+1 SAVE IR1                      CSP05010
0002 01 65800000          LDX     I1 DECA1 PUT ARGUMENT ADDRESS IN IR1    CSP0502D
0004 0  C100              LD      1 0    GET JCARD ADDRESS                CSP05030
0005 0  D039              STO       JCRD1 SETUP JCARD ADDRESS FOR NZONE   CSP05040
0006 00 95800002    TWO   S       I1 2    SUBTRACT JLAST VALUE            CSP05050
0008 0  D020              STO       PICK+1 PLACE LOAD ADDRESS FOR CONVRSN CSP05060
0009 0  D030              STO       PUT+1 PLACE STORE ADDRESS FOR CONVRSN CSP05070
000A 0  8007              A         ONE+1 ADD CONSTANT OF ONE             CSP05080
000B 0  D010              STO       TEST+1 CREATE JCARD(JLAST) ADDRESS    CSP05090
000C 0  C102              LD      1 2    GET JLAST ADDRESS               CSP05100
000D 0  DD32              STO       JLAS1 SETUP JLAST ADDRESS FOR NZONE   CSP05110
000E 01 C4800D40          LD      I JLAS1 GET JLAST VALUE AND             CSP05120
0010 0  DDEF              STO       DECA1 SAVE IT AT DECA1                CSP05130
0011 D0 958000D1    ONE   S       I1 1    SUBTRACT J VALUE                CSP0514D
0013 0  80FE              A         ONE+1 ADD CONSTANT OF ONE             CSPD5150
0014 0  4808              BSC       +     CHECK FIELD WIDTH               CSP05160
0015 0  CDFC              LD        ONE+1 NEGATIVE OR ZERO-MAKE IT ONE    CSP0517D
0016 0  D010              STO       COUNT+1 OK-SAVE WIDTH IN COUNT        CSP05180
0017 0  C103              LD      1 3    GET NER ADDRESS                 CSP0519D
0018 0  D018              STO       ERA+1 SAVE IT                         CSP05200
0019 0  7104              MDX     1 4    MOVE OVER FOUR ARGUMENTS         CSP05210
001A 0  692B              STX     1 DONE1+1 CREATE RETURN ADDRESS         CSP05220
                     *                 CHECK THE SIGN OF JCARD.  IF      CSP05230
                     *                 NEGATIVE, SET JSIGN=2, AND MAKE   CSP05240
                     *                 IT POSITIVE,  OTHERWISE, SET      CSP05250
                     *                 JSIGN=4                           CSP05260
001B 00 C4000000    TEST  LD      L *-*  GET JCARD(JLAST)                 CSP05270
001D 01 4C280021          BSC     L NEG,+Z CHECK FOR NEGATIVE             CSP05280
001F 0  C027              LD        FOUR NO - LOAD FOUR                   CSP05290
0020 0  7004              MDX       GO    SKIP OVER NEGATIVE PROCESSING   CSP05300
0021 0  F026        NEG   EOR       HFFFF YES - CHANGE SIGN TO POSITIVE   CSP05310
0022 01 D480001C          STO     I TEST+1 RESTORE SIGN AS POSITIVE       CSP05320
0024 0  C0E2              LD        TWO+1 LOAD TWO                        CSP05330
0025 0  D0F6        GO    STO       TEST+1 STORE ACCUMULATOR TO SAVE SIGN CSP05340
                     *                 JNOW=J                            CSP05350
0026 00 65000000    COUNT LDX     L1 *-*  LOAD IR1 WITH FIELD WIDTH       CSPD5360
                     *                 JTEST=JCARD(JNOW)                 CSPD5370
0028 00 C5000000    PICK  LD      L1 *-*  PICK UP JCARD(JNOW)             CSP05380
002A 01 4C100D33          BSC     L OK,- AND CHECK IT AGAINST ZERO        CSP05390
                     *                 NER=JNOW                          CSP05400
002C 0  69FA        ERR   STX       1 COUNT+1 LESS THAN - ERROR           CSP05410
002D 0  CDD2              LD        DECA1 CALCULATE THE SUBSCRIPT         CSP05420
002E D  9DF8              S         COUNT+1 OF THIS DIGIT                 CSP05430
002F 0  80E2              A         ONE+1 AND STORE                       CSP05440
0030 00 D4000000    ERA   STO     L *-*  IT AT NER                        CSP05450
```



```
0032 0  7008              MDX       MORE GET NEXT DIGIT                   CSP05460
0033 0  9015        OK    S         TEN  NOT LESS - COMPARE IT TO         CSP05470
D034 01 4C10002C          BSC     L ERR,- CONSTANT OF TEN-NOT LESS GO TO  CSP05480
                     *                 ERR                               CSP05490
D036 0  8D12              A         TEN  LESS - ADD TEN BACK              CSP05500
0037 0  1008              SLA       8    SHIFT THE FOUR BITS OF DECIMAL   CSP05510
0038 0  E811              OR        ZERO IN PLACE AND CREATE A1           CSP05520
D039 00 05000000    PUT   STO     L1 *-*  CHARACTER-STORE IN JCARD(JNOW)  CSP05530
                     *                 SEE IF JNOW IS LESS THAN JLAST.   CSP05540
                     *                 IF YES, JNOW=JNOW+1 AND GO BACK   CSP05550
                     *                 FOR MORE.  IF NO, SETUP THE SIGN  CSP05560
003B 0  71FF        MORE  MDX       1 -1   DECREMENT THE FIELD WIDTH      CSP05570
003C 0  70EB              MDX       PICK GO BACK FOR MORE                 CSPD5580
0030 30 15A56545          CALL      N2ONE N2ONE ROUTINE TO PLACE SIGN     CSP05590
003F 0  0000        JCRD1 DC        *-*  ADDRESS OF JCARD                 CSP05600
0040 0  0000        JLAS1 DC        *-*  ADDRESS OF JLAST                 CSP05610
0041 1  001C              DC        TEST+1 ADDRESS OF SIGN INDICATOR TO   CSP05620
                     *                 USE                               CSP0563D
0042 1  003F              DC        JCRD1 ADDRESS OF SIGN INDICATOR FOR   CSP05640
                     *                 OLD SIGN                          CSP05650
                     *                 EXIT                              CSP05660
0043 00 65000000    SAVE1 LDX     L1 *-*  RESTORE IR1                     CSP05670
0045 00 4C000000    DONE1 BSC     L  *-*  RETURN TO CALLING PROGRAM       CSP05680
0047 0  0D04        FOUR  DC        4    CONSTANT OF FOUR                 CSP05690
0048 D  FFFF        HFFFF DC        /FFFF CONSTANT OF ALL BINARY ONES     CSP05700
0049 0  D00A        TEN   DC        10   CONSTANT OF TEN                  CSPD5710
004A 0  F040        ZERO  DC        /F04D CONSTANT OF EBCDIC ZERO         CSPD572D
004C                      END                                             CSP05730
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP05740
*STORE      WS  UA  DECA1                                              CSP05750
3345 0006
```

```
// ASM                                                                  CSP05760
** DIV SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE                (ID) CSP05770
* NAME DIV                                                              (ID) CSP05780
* LIST                                                                       CSP05790
0000    04265000          ENT     DIV  DIVIDE SUBROUTINE ENTRY POINT       CSP05800
                *             CALL DIV(JCARO,J,JLAST,KCARD,K,KLAST,NER)  CSP05810
                *                  THE WORDS JCARD(J) THROUGH            CSP05820
                *                  JCARD(JLAST) ARE DIVIDED INTO         CSP05830
                *                  THE WORDS KCARD(K) THROUGH            CSP05840
                *                  KCARD(KLAST).  THE KCARO FIELD        CSP05850
                *                  IS EXTENDED TO THE LEFT AND           CSP05860
                *                  CONTAINS THE QUOTIENT AND             CSP05870
                *                  REMAINDER.                            CSP05880
0000 0  0000    DIV   DC      *-*  ARGUMENT ADDRESS COMES IN HERE          CSP05890
0001 0  697D          STX   1 SAVE1+1 SAVE IR1                           CSP05900
0002 0  6A71          STX   2 SAVE2+1 SAVE IR2                           CSP05910
0003 0  6B72          STX   3 SAVE3+1 SAVE IR3                           CSP05920
0004 01 65800000      LDX   I1 DIV  PUT ARGUMENT ADDRESS IN IR1          CSP05930
0006 0  C100          LD    1 0     GET JCARD ADDRESS                    CSP05940
0007 00 95B00002      S     I1 2    SUBTRACT JLAST VALUE                 CSP05950
0009 0  D04C          STO     SRCH+1 STORE END OF JCARD ADDRESS          CSP05960
000A 01 D40000AD      STO   L MULT1+1 FOR SEARCH AND MULTIPLICATION      CSP05970
000C 0  B004          A       ONE+1 ADD CONSTANT OF ONE                  CSP05980
000D 0  D011          STO     SGNJ+1 CREATE JCARD(JLAST) ADDRESS         CSP05990
                *             JSPAN=JLAST-J+1                            CSP06000
000E 00 C5800002  TWO   LD    I1 2    GET JLAST VALUE                    CSP06010
0010 00 95B00001  ONE   S     I1 1    SUBTRACT J VALUE                   CSP06020
0012 0  80FE          A       ONE+1 ADD CONSTANT OF ONE                  CSP06030
0013 0  480B          BSC     +     CHECK FIELD WIDTH                    CSP06040
0014 0  C0FC          LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE         CSP06050
0015 0  D03E          STO     SRCHT+1 STORE COUNT FOR SEARCH             CSP06060
0016 0  C103          LO    1 3     GET KCARD ADDRESS                    CSP06070
0017 0  D037          STO     KCRD1 SAVE FOR FILL                        CSP06080
001B 00 95800005      S     I1 5    SUBTRACT KLAST VALUE                 CSP06090
001A 0  80F6          A       ONE&1 ADD CONSTANT OF ONE                  CSP06100
001B 0  D00D          STO     SGNK+1 CREATE KCARD(KLAST) ADDRESS         CSP06110
001C 0  7107          MDX   1 7     MOVE OVER SEVEN ARGUMENTS            CSP06120
001D 0  695A          STX   1 DONE1+1 CREATE RETURN ADDRESS              CSP06130
                *                   CLEAR AND SAVE THE SIGNS ON THE    CSP06140
                *                   JCARD AND THE KCARD FIELDS         CSP06150
001E 00 C4000000  SGNJ  LD    L *-*  PICKUP THE SIGN OF JCARD            CSP06160
0020 0  D00F          STO     DIV  SAVE IT IN DIV                        CSP06170
0021 01 4C100027      BSC   L JPLUS,- IF NOT NEGATIVE-GO TO JPLUS        CSP06180
0023 0  F039          EOR     HFFFF+1 NEGATIVE-MAKE IT POSITIVE          CSP06190
0024 01 D480001F      STO   I SGNJ+1 PUT BACK IN JCARD(JLAST)            CSP06200
0026 0  C036          LD      HFFFF+1 LOAD A MINUS ONE                   CSP06210
0027 0  1890      JPLUS SRT     16    SAVE IN EXTENSION                  CSP06220
002B 00 C4000000  SGNK  LD    L *-*  PICKUP THE SIGN OF KCARD            CSP06230
002A 0  D04F          STO     KSIGN SAVE IT IN KSIGN                     CSP06240
002B 01 4C100033      BSC   L KPLUS,- IF NOT NEGATIVE-GO TO KPLUS        CSP06250
002D 0  F02F          EOR     HFFFF+1 NEGATIVE-MAKE IT POSITIVE          CSP06260
002E 01 D4B00029      STO   I SGNK+1 PUT BACK IN KCARD(KLAST)            CSP06270
0030 0  1090          SLT     16    GET SIGN OF JCARD                    CSP06280
0031 0  F02B          EOR     HFFFF+1 CHANGE IT                          CSP06290
0032 0  7001          MDX     OVRK SKIP NEXT INSTRUCTION                 CSP06300
0033 0  1090      KPLUS SLT     16    GET SIGN OF JCARD                  CSP06310
0034 0  D046      OVRK  STO     QSIGN STORE FOR SIGN OF QUOTIENT           CSP06320
```

```
                        *                       KSTRT=K-1                                   CSP06330
0035 00 C580FFFD                LD    I1 -3     GET VALUE OF K                              CSP06340
0037 0  8025                    A       HFFFF61 SUBTRACT CONSTANT OF ONE                    CSP06350
0038 0  0040                    STO     KSTRT SAVE IN KSTRT                                 CSP06360
                        *                       KLOW=K-JSPAN                                CSP06370
0039 0  8007                    A       ONE+1 GET VALUE OF K                                CSP06380
003A 0  9019                    S       SRCHT+1 SUBTRACT JSPAN                              CSP06390
003B 0  0041                    STO     KLOW SAVE IN KLOW                                   CSP06400
003C 00 C580FFFE        MTWO    LD    I1 -2     GET KLAST VALUE                             CSP06410
003E 0  D040                    STO     TMP   SAVE IT                                       CSP06420
                        *                       CALCULATE THE ADDRESS OF THE                CSP06430
                        *                       SIGN OF THE QUOTIENT                        CSP06440
003F 0  C00F                    LD      KCRD1 GET KCARD ADDRESS                             CSP06450
0040 0  903E                    S       TMP   SUBTRACT KLAST VALUE                          CSP06460
0041 0  8012                    A       SRCHT+1 ADD JSPAN                                   CSP06470
0042 0  80CE                    A       ONE+1 ADD CONSTANT OF ONE                           CSP06480
0043 01 040000DF                STO   L OUOT+1 STORE ADDR OF SIGN OF QUOTIENT               CSP06490
                        *                       IS KLAST-KSTRT-JSPAN NEGATIVE               CSP06500
0045 0  C039                    LD      TMP   LOAD KLAST VALUE                              CSP06510
0046 0  9032                    S       KSTRT SUBTRACT KSTRT                                CSP06520
0047 0  900C                    S       SRCHT+1 SUBTRACT JSPAN                              CSP06530
0048 01 4C28005B                BSC   L ERR,+Z IF NEGATIVE-GO TO ERROR                      CSP06540
                        *                       IS KLOW POSITIVE                            CSP06550
004A 0  C032                    LO      KLOW OK-GET KLOW VALUE                              CSP06560
004B 01 4C08005B                BSC   L ERR,+ IF NOT POSITIVE-GO TO ERROR                   CSP06570
                        *                       FILL THE EXTENSION OF KCARD WITH            CSP06580
                        *                       ZEROES                                      CSP06590
0040 30 062534C0                CALL    FILL  OK-FILL EXTENSION WITH ZEROES               CSP06600
004F 0  0000            KCR01   OC      *-*   ADDRESS OF KCARD                            CSP06610
0050 1  0070                    OC      KLOW  ADDRESS OF LEFT END OF EXTENSION            CSP06620
0051 1  0079                    OC      KSTRT ADDRESS OF RGHT END OF EXTENSON             CSP06630
0052 1  007C                    OC      ZIP   ADDRESS OF CONSTANT OF ZERO                 CSP06640
                        *                       JFRST=J                                     CSP06650
0053 00 66000000        SRCHT   LOX   L2 *-*  LOAD IR2 WITH JCARD COUNT                    CSP06660
0055 00 C6000000        SRCH    LD    L2 *-*  PICKUP JCARD(JFRST)                          CSP06670
                        *                       IS JCARD(JFRST) POSITIVE                    CSP06680
0057 01 4C300080                BSC   L HIT,-Z IF POSITIVE-GO TO HIT                      CSP06690
                        *                       SEE IF JFRST IS LESS THAN JLAST.            CSP06700
                        *                       IF YES, JFRST=JFRST+1 AND GO                CSP06710
                        *                       BACK FOR MORE.  IF NO, ERROR.               CSP06720
0059 0  72FF                    MOX     2 -1  DECREMENT IR2                               CSP06730
005A 0  70FA                    MOX     SRCH  GO BACK FOR MORE                            CSP06740
                        *                       ERROR - NER=KLAST                           CSP06750
005B 0  C023            ERR     LD      TMP   PICKUP KLAST VALUE                          CSP06760
005C 00 0580FFFF        HFFFF   STO   I1 -1   AND STORE IN NER                            CSP06770
                        *                       REPLACE JCARD SIGN                          CSP06780
005E 0  C0A1            FINER   LD      QIV   PICKUP JCARD SIGN AND                       CSP06790
005F 01 0480001F                STO   I SGNJ+1 PUT IT BACK                                CSP06800
                        *                       REPLACE KCARD SIGN                          CSP06810
0061 0  C018                    LD      KSIGN PICKUP KCARD SIGN                           CSP06820
0062 01 4C28006C                BSC   L KNEG,+Z IF NEGATIVE-GO TO KNEG                    CSP06830
0064 01 C4800029                LO    I SGNK+1 NOT NEGATIVE-PICKUP NEW SIGN               CSP06840
0066 01 4C100071                BSC   L SAVE1,- IF NOT NEGATIVE-GO TO EXIT                CSP06850
0068 0  F0F4            BCK1    EOR     HFFFF+1 NEGATIVE-CHANGE SIGN AND                  CSP06860
0069 01 D4800029                STO   I SGNK+1 PUT INTO KCARD(KLAST)                      CSP06870
```

```
006B 0  7005              MDX     SAVE1 GO TO EXIT                          CSP06880
006C 01 C4800029  KNEG    LD    I SGNK+1 NEGATIVE=PICKUP NEW SIGN           CSPD6890
006E 01 4C280071          BSC   L SAVE1,+2 IF NEGATIVE=GO TO EXIT           CSP069DD
0070 0  70F7              MDX     BCK1 NOT NEGATIVE=GO TO BCK1              CSP06910
                *                       EXIT.............................. CSP0692D
0071 00 65000000  SAVE1 LDX   L1 *-*  RESTORE IR1                          CSP06930
0073 00 66000000  SAVE2 LDX   L2 *-*  RESTORE IR2                          CSPD694D
0075 00 67000000  SAVE3 LDX   L3 *-*  RESTORE IR3                          CSPD6950
0077 00 4C000000  DONE1 BSC   L  *-*  RETURN TO CALLING PROGRAM             CSP0696D
0079 0  0000      KSTRT DC        *-*  ONE LESS THAN K                      CSP0697D
007A 0  0000      KSIGN DC        *-*  SIGN OF KCARD                        CSP06980
007B 0  0000      QSIGN DC        *-*  SIGN OF QUOTIENT                     CSP0699D
007C 0  0000      ZIP   DC        0    CONSTANT OF ZERO                     CSP07D00
0070 0  0000      KLOW  DC        *-*  SUBSCRIPT OF LEFTMOST POSITION       CSP0701D
                *                       OF EXTENSION OF KCARD             CSP07D20
007E 0  000A      TEN   DC        10   CONSTANT OF TEN                      CSP07030
007F 0  0000      TMP   DC        *-*  TEMPORARY STORAGE                    CSP07040
                *                       JHIGH=JCARD(JFRST)                CSP07050
0080 0  D0D3      HIT   STO       SRCHT+1 SAVE FIRST SIGNIFICANT DIGIT      CSP0706D
                *                       KPUT=KLOW+JLAST-JFRST             CSP07070
0081 0  6A28              STX   2 JLOOP+1 GET THE VALUE OF JLAST-JFRST      CSP07080
0082 0  CDCC              LD      KCRD1 GET KCARD ADDRESS                   CSP07090
0083 0  D03E              STO     KCRD2 SAVE FOR CARRY                      CSP0710D
0084 0  90F8              S       KLOW SUBTRACT KLOW VALUE                  CSP07110
0085 0  9024              S       JLOOP+1 SUBTRACT JLAST=JFRST VALUE        CSP07120
0086 0  90B6              S       MTWO+1 ADD CONSTANT OF TWO                CSP07130
0087 0  D04E              STO     PUT2+1 SAVE ADDRESS FOR STORING           CSP07140
                *                       KSTOP=KLAST+JFRST-JLAST-1         CSP07150
0088 0  C0F6              LD      TMP  GET KLAST VALUE                      CSP07160
0089 0  9020              S       JLOOP+1 SUBTRACT JLAST=JFRST VALUE        CSP07170
008A 0  90D2              S       HFFFF+1 ADD CONSTANT OF ONE               CSP0718D
008B 0  D0CA              STO     SRCH61 SAVE VALUE FOR COMPLIMENTING       CSP07190
008C 0  90EC              S       KSTRT SUBTRACT KSTRT VALUE                CSP07200
008D 0  D00B              STO     LOOPM+1 SAVE COUNT AT LOOPM+1             CSP07210
008E 0  C033              LD      KCRD2 GET KCARD ADDRESS                   CSP07220
008F 0  90EF              S       TMP  SUBTRACT KLAST VALUE                 CSP0723D
0090 0  8019              A       JLOOP61 ADD JLAST=JFRST VALUE             CSP07240
0091 0  D009              STO     DIV161 SAVE FOR MULT. BY TEN              CSP0725D
0092 0  D038              STO     DIV561 SAVE FOR ADD OF 10*KNOW            CSP07260
0093 0  D039              STO     DIV661 SAVE FOR STORE OF 1D*KNOW          CSP07270
0094 D  80C8              A       HFFFF+1 SUBTRACT CONSTANT OF ONE          CSP07280
0095 0  D009              STO     DIV261 SAVE FOR ADD INTO MULT             CSP0729D
0096 0  D01A              STO     DIV361 SAVE FOR SUBTRACTION FROM          CSP07300
0097 0  D01B              STO     DIV461 SAVE FOR STORE SUBTRACTED FROM     CSPD731D
                *                       KM=KSTRT                          CSP0732D
0098 00 65000000  LOOPM LDX   L1 *-*  LOAD IR1 WITH COUNT                   CSP0733D
                *                       MULT=(10*KCARD(KM)+KCARD(KM+1))   CSP0734D
                *                                 DIVIDED BY JHIGH        CSP07350
009A 00 C5000000  DIV1  LD    L1 *-*  PICKUP KCARD(KM)                    CSP07360
009C 0  A0E1              M       TEN  MULTIPLY BY TEN                      CSPD737D
009D 0  1090              SLT     16   REPOSITION PRODUCT                   CSPD738D
009E 00 85000D00  DIV2  A     L1 *-*  ADD IN KCARD(KM+1)                    CSP0739D
00A0 0  1890              SRT     16   REPOSITION FOR DIVISION              CSP07400
00A1 0  A8B2              D       SRCHT+1 DIVIDE BY JHIGH                   CSP0741D
00A2 0  D0DA              STO     KLOW SAVE IN KLOW(MULT)                   CSP0742D
```

```
                        *                       NOUO=MULT                                CSP07430
00A3 0  D0D5                    STO     KSTRT SAVE IN KSTRT(NOUO)                    CSP07440
                        *                       IS MULT GREATER THAN ZERO            CSP07450
00A4 01 4C0800D4                BSC  L  PUT,+ IF MULT NOT POSITIVE-GO TO PUT         CSP07460
                        *                       KNOW=KM+1                            CSP07470
00A6 0  6901            ADBCK   STX    1 KNOW+1 POSITIVE-GET KM+1 AND                CSP07480
00A7 00 67000000        KNOW    LDX  L3 *-*  PUT IT IN IR3                           CSP07490
                        *                       JNOW=JFRST                           CSP07500
00A9 00 66000000        JLOOP   LDX  L2 *-*  RELOAD IR2 WITH REMAINING JCARD         CSP07510
00AB 0  1810                    SRA     16   CLEAR ACCUMULATOR                       CSP07520
                        *                       KCARD(KNOW)=KCARD(KNOW)    -         CSP07530
                        *                                  MULT*JCARD(JNOW)          CSP07540
00AC 00 96000000        MULT1   S    L2 *-*  LOAD NEGATIVE JCARD(JNOW)               CSP07550
00AE 0  A0CE                    M       KLOW MULTIPLY BY MULT                        CSP07560
00AF 0  1090                    SLT     16   REPOSITION PRODUCT                      CSP07570
00B0 00 87000000        DIV3    A    L3 *-*  ADD IN KCARD(KNOW)                       CSP07580
00B2 00 D7000000        DIV4    STO  L3 *-*  STORE AT KCARD(KNOW)                    CSP07590
                        *                       KNOW=KNOW+1                          CSP07600
00B4 0  73FF                    MDX    3 -1   DECREMENT IR3                         CSP07610
00B5 0  7000                    MDX     *     NOP                                    CSP07620
                        *                       IS JNOW LESS THAN JLAST, IF YES      CSP07630
                        *                       JNOW=JNOW+1 AND GO BACK FOR MORE     CSP07640
                        *                       IF NO, RESOLVE CARRIES,              CSP07650
00B6 0  72FF                    MDX    2 -1   DECREMENT IR2                         CSP07660
00B7 0  70F3                    MDX     JLOOP+2 NOT DONE-GO BACK FOR MORE           CSP07670
00B8 0  69EF                    STX    1 KNOW+1 DONE-CALCULATE                       CSP07680
00B9 0  C09C                    LD      SRCH61 THE VALUE OF                         CSP07690
00BA 0  90ED                    S       KNOW+1 KNOW-1                               CSP07700
00BB 0  D0EC                    STO     KNOW+1 BY COMPLIMENTING COUNT               CSP07710
00BC 0  6BDC                    STX    3 LOOPM+1 CALCULATE THE                       CSP07720
00BD 0  C098                    LD      SRCH61 VALUE OF KM                          CSP07730
00BE 0  90DA                    S       LOOPM+1 BY COMPLIMENTING THE                CSP07740
00BF 0  D0D9                    STO     LOOPM+1 OTHER COUNT                         CSP07750
                        *                       RESOLVE CARRIES IN THIS RESULT       CSP07760
00C0 30 03059668                CALL    CARRY RESOLVE CARRIES                       CSP07770
00C2 0  0000            KCRD2   DC      *-*  ADDRESS OF KCARD                       CSP07780
00C3 1  00A8                    DC      KNOW+1 ADDRESS OF KM                        CSP07790
00C4 1  0099                    DC      LOOPM+1 ADDRESS OF KNOW-1                   CSP07800
00C5 1  00A8                    DC      KNOW+1 ADDRESS OF GENERATED CARRY           CSP07810
                        *                       IS KNOW LESS THAN ZERO               CSP07820
00C6 01 4C1000D4                BSC  L  PUT,- IF NOT NEGATIVE-GO TO PUT             CSP07830
                        *                       KCARD(KM)=KCARD(KM)+1D*KNOW          CSP07840
00C8 0  A0B5                    M       TEN  NEGATIVE-MULTIPLY CARRY BY TEN         CSP07850
00C9 0  1090                    SLT     16   REPOSITION PRODUCT                      CSP07860
00CA 00 85000000        DIV5    A    L1 *-*  ADD IN KCARD(KNOW)                       CSP07870
00CC 00 D5000000        DIV6    STO  L1 *-*  STORE AT KCARD(KNOW)                    CSP07880
                        *                       MULT=-1                              CSP07890
00CE 0  C08E                    LD      HFFFF+1 LOAD A MINUS ONE                    CSP07900
00CF 0  D0AD                    STO     KLOW STORE IN MULT                          CSP07910
                        *                       NOUO=NOUO-1                          CSP07920
00D0 0  C0A8                    LD      KSTRT LOAD THE VALUE OF NOUO                CSP07930
00D1 0  808B                    A       HFFFF+1 SUBTRACT CONSTANT OF ONE            CSP07940
00D2 0  D0A6                    STO     KSTRT STORE IN NOUO                         CSP07950
00D3 0  70D2                    MDX     ADBCK GO TO ADD OVERDRAW BACK               CSP07960
                        *                       KCARD(KPUT)=NOUO                     CSP07970
```



```
00D4 0  C0A4            PUT     LD      KSTRT LOAD NOUO                             CSP07980
00D5 00 D4000000        PUT2    STO  L  *-*  STORE AT KCARD(KPUT)                   CSP07990
                        *                       KPUT=KPUT+1                          CSP08000
00D7 01 74FF00D6                MDX  L  PUT2+1,=-1 MODIFY KCARD(KPUT) ADDRESS       CSP08010
                        *                       SEE IF KM IS LESS THAN KSTOP,        CSP08020
                        *                       IF YES, KM=KM+1 AND GO BACK FOR      CSP08030
                        *                       MORE,  IF NO, PLACE ALL SIGNS,       CSP08040
00D9 0  71FF                    MDX    1 -1   DECREMENT IR1                         CSP08050
00DA 0  70BF                    MDX     DIV1 NOT DONE-GO BACK FOR MORE              CSP08060
                        *                       PUT SIGN ON QUOTIENT                 CSP08070
00DB 0  C09F                    LD      OSIGN DONE-PICKUP SIGN OF QUOTIENT          CSP08080
00DC 01 4C2800E8                BSC  L  NEG,+Z IF NEGATIVE-GO TO NEG                CSP08090
00DE 00 C4000000        QUOT    LD   L  *-*  NOT NEGATIVE-PICKUP ACTUAL SIGN        CSP08100
00E0 01 4C10005E                BSC  L  FINER,- IF NOT NEGATIVE-GO TO OTHERS        CSP08110
00E2 01 F400005D        BCK2    EOR  L  HFFFF+1 NEGATIVE-CHANGE SIGN                CSP08120
00E4 01 D48000DF                STO  I  QUOT+1 PUT SIGN ON QUOTIENT                 CSP08130
00E6 01 4C00005E                BSC  L  FINER, GO TO REPLACE OTHER SIGNS            CSP08140
00E8 01 C48000DF        NEG     LD   I  QUOT+1 NEGATIVE-PICKUP ACTUAL SIGN          CSP08150
00EA 01 4C28005E                BSC  L  FINER,+Z IF NEGATIVE-GO TO OTHER SIGN       CSP08160
00EC 0  70F5                    MDX     BCK2 GO TO CHANGE SIGN                      CSP08170
00EE                            END                                                 CSP08180
```

NO ERRORS IN ABOVE ASSEMBLY,

```
// DUP                                                                              CSP0B190
*STORE      WS  UA  DIV                                                             CSP08200
334B 000F
```

```
// ASM                                                                  CSP08210
** DPACK/OUNPK SUBROUTINES FOR 1130 COMMERCIAL SUBROUTINE PACKAGE  (IO) CSP08220
* NAME OUNPK                                                           (IO) CSP08230
* LIST                                                                 CSP08240
0000    04915502          ENT     DUNPK OUNPK SUBROUTINE ENTRY POINT   CSP08250
                  *               CALL OUNPK(JCARO,J,JLAST,KCARD,K)    CSP08260
                  *               THE WORDS JCARD(J) THROUGM           CSP08270
                  *               JCARD(JLAST) IN 04 FORMAT ARE        CSP08280
                  *               UNPACKED INTO KCARO IN 01 FORMAT.    CSP08290
0006    045CI0D2          ENT     OPACK OPACK SUBROUTINE ENTRY POINT   CSP08300
                  *               CALL OPACK(JCARO,J,JLAST,KCARD,K)    CSP08310
                  *               TME WOROS JCARO(J) TMROUGM           CSP08320
                  *               JCARO(JLAST) IN 01 FORMAT ARE PACKED CSP08330
                  *               INTO KCARO IN D4 FORMAT.             CSP08340
0000 0  0000      DUNPK DC        *-* ARGUMENT ADDRESS COMES IN HERE   CSP08350
0001 0  C003            LD      SW2 LOAD NOP INSTRUCTION             CSP08360
0002 0  D020            STO     SWTCM STORE NOP AT SWITCM            CSP08370
0003 0  7007            MOX     START COMPUTING                      CSP08380
0004 0  7027      SW1   MOX   X ELSE-SWTCM-1 BRANCH TO ELSE          CSP08390
0005 0  7000      SW2   MOX   X 0  NOP INSTRUCTION                   CSP08400
0006 0  0000      OPACK OC        *-* ARGUMENT ADDRESS COMES IN HERE   CSP08410
0007 0  C0FE            LO      DPACK PICK UP ARGUMENT ADDRESS       CSP08420
0008 0  00F7            STO     OUNPK AND STORE IT IN OUNPK          CSP08430
0009 0  C0FA            LD      SW1 LOAD BRANCH TO ELSE              CSP08440
000A 0  D018            STO     SWTCM STORE BRANCH AT SWITCH         CSP08450
000B 0  6952      START STX     1 SAVE1+1 SAVE IR1                   CSP08460
000C 0  6A53            STX     2 SAVE2+1 SAVE IR2                   CSP08470
000D 01 65800000        LOX   I1 OUNPK PUT ARGUMENT ADDRESS IN IRI   CSP08480
000F 0  C100            LO    1 0 GET JCARO ADDRESS                  CSP08490
0010 0  8001            A       ONE+1 ADD CONSTANT OF 1              CSP08500
0011 00 95800001  ONE   S     I1 I SUBTRACT J VALUE                  CSP08510
0013 0  D00D            STO     JCARO+1 CREATE JCARO(J) ADDRESS      CSP08520
0014 0  CI03            LD    1 3 GET KCARO ADDRESS                  CSP08530
0015 0  80FC            A       ONE+1 ADD CONSTANT OF 1              CSP08540
0016 00 95800004  FOUR  S     II 4 SUBTRACT K VALUE                  CSP08550
0018 0  0006            STO     KCARO+1 CREATE KCARO(K) ADDRESS      CSP08560
0019 0  C100            LO    1 0 GET JCARO ADDRESS                  CSP08570
001A 0  80F7            A       ONE+I ADD CONSTANT OF 1              CSP08580
001B 00 95800002        S     I1 2 SUBTRACT JLAST VALUE              CSP08590
0010 0  D0E8            STO     OPACK CREATE JCARD(JLAST) ADDRESS    CSP08600
001E 00 65000000  KCARO LOX   LI *-* PUT KCARO ADDRESS IN IR1         CSP08610
0020 00 C4000000  JCARO LO    L *-* PICK UP JCARO(J)                 CSP08620
0022 0  6204            LOX   2 4     LOAD IR2 WITM 4, DIGITS/WORD   CSP08630
0023 0  7000      SWTCH MDX   X 0 SWITCH BETWEEN DPACK AND OUNPK      CSP08640
0024 0  1890            SRT     16    TEMPORARILY SAVE ACCUM IN EXTNTN CSP08650
                  *               CHECK FOR JCARD(JLAST)               CSP08660
0025 0  C0FB            LO      JCARO+1 PICK UP CURRENT JCARO ADDR   CSP08670
0026 0  90DF            S       OPACK SUBTRACT JCARO(JLAST)          CSP08680
0027 01 4C080059        BSC   L ALLOO,+ IF ZERO, ALL ONE - ALLDO     CSP08690
0029 0  1810      AGAIN SRA     16    NOT DONE - CLEAR ACCUMULATOR   CSP08700
002A 0  1084            SLT     4     GET FIRST DIGIT OF WORD        CSP08710
002B 0  F00A            EOR     H000F IS IT FILLER                   CSP08720
002C 01 4C180031        BSC   L NEXT,+- YES - GO TO NEXT             CSP08730
002E 0  F007            EOR     H000F NO - RESTORE TO ORIGINAL       CSP08740
002F 0  D100            STO   1 0     STORE IN KCARD                 CSP08750
0030 0  71FF            MDX   1 -1    GO TO NEXT WORD OF KCARO       CSP08760
0031 0  72FF      NEXT  MOX   2 -1    DECREMENT DIGITS/WORD          CSP08770
```



```
0032 0  70F6            MOX     AGAIN MORE IN THIS WORD - GO BACK    CSP08780
0033 01 74FF0021        MOX   L JCARO+1,-1 THIS WORD ONE              CSP08790
                  *               GET NEXT WORD IN JCARO               CSP08800
0035 0  70EA            MOX     JCARO GO BACK                        CSP08810
0036 0  000F      H000F DC        /000F CONSTANT OF 15 TO DETECT FILLER CSP08820
0037 01 74010021  EN    MOX   L JCARO+1,1 BACK UP JCARO FOR SIGN      CSP08830
0039 0  6AE5            STX   2 KCARO+1 IF DIGITS/WORD IS FOUR,      CSP08840
003A 0  C0E4            LO      KCARO+1 ALL ONE EXCEPT FOR SIGN      CSP08850
003B 0  90DB            S       FOUR+1 SUBTRACT FOUR FROM DIGITS/WORD CSP08860
003C 01 4C180046        BSC   L LAST,+- IF ZERO - ALL ONE - GO LAST  CSP08870
003E 0  1884            SRT     4     NOT ONE - TAKE OUT SIGN        CSP08880
003F 0  C023      BACK  LO        HF000 PUT IN FILLER                  CSP08890
0040 0  180C            RTE     28    SET FILLER IN LOW ORDER OF EXTN CSP08900
0041 0  72FF            MOX   2 -1    DECREMENT DIGITS/WORD          CSP08910
0042 0  70FC            MOX     BACK MORE - GO BACK                  CSP08920
0043 0  1090            SLT     16    ONE - PUT EXTENSION IN ACCUM   CSP08930
0044 0  0100            STO   1 0     STORE IN KCARD                 CSP08940
0045 0  71FF            MOX   1 -1    GET NEXT WORD OF KCARO FOR SIGN CSP08950
0046 01 C4800021  LAST  LO    I JCARD+I PICK UP SIGN OF JCARO        CSP08960
0048 0  7011            MDX     ALLOO+1 GO TO INSTRUCTION AFTER ALLOO CSP08970
0049 01 C4800021  OVR   LO    I JCARO+1 PICK UP NEXT JCARO DIGIT     CSP08980
004B 0  I00C      ELSE  SLA     12    PUT DIGIT IN HIGH ORDER OF ACC CSP08990
004C 0  180C            RTE     28    SET DIGIT IN LOW ORDER OF EXTN CSP09000
0040 01 74FF0021        MOX   L JCARO+1,-1 GET NEXT JCARO WORD       CSP09010
                  *               CHECK FOR JCARO(JLAST)               CSP09020
004F 0  C00I            LO      JCARO+1 PICK UP CURRENT JCARO ADDR   CSP09030
0050 0  90B5            S       OPACK SUBTRACT JCARO(JLAST)          CSP09040
0051 01 4C280037        BSC   L EN,+2 IF ZERO,ALL ONE - GO TO EN     CSP09050
0053 0  72FF            MOX   2 -1    NOT ONE-DECREMENT DIGITS/WORD  CSP09060
0054 0  70F4            MOX     OVR  GO BACK FOR NEXT DIGIT          CSP09070
0055 0  1090            SLT     16    WORD FULL-PUT EXTN IN ACCUM    CSP09080
0056 0  0100            STO   1 0     STORE IN KCARD                 CSP09090
0057 0  71FF            MOX   1 -1    GET NEXT KCARO WORD            CSP09100
0058 0  70C7            MOX     JCARO GO BACK                        CSP09110
0059 0  1090      ALLOO SLT     I6    ONE-PUT EXTENSION IN ACCUMULTR CSP09120
005A 0  0100            STO   1 0     STORE SIGN IN KCARO            CSP09130
005B 01 74050000        MOX   L OUNPK,5 CREATE RETURN ADDRESS        CSP09140
0050 00 65000000  SAVE1 LOX   L1 *-* RESTORE IR1                     CSP09150
005F 00 66000000  SAVE2 LOX   L2 *-* RESTORE IR2                     CSP09160
0061 01 4C800000        BSC   I OUNPK RETURN TO CALLING PROGRAM      CSP09170
0063 0  F000      MF000 OC        /F000 CONSTANT OF 15 FOR FILLER      CSP09180
0064                    END                                          CSP09190

    NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                                  CSP09200
*STORE      WS  UA  DUNPK                                               CSP09210
335A 0007
```

```
// ASM                                                                  CSP09220
** EDIT SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP09230
* NAME EDIT                                                        (ID) CSP09240
* LIST                                                                 CSP09250
0000    051098C0          ENT     EDIT EDIT SUBROUTINE ENTRY POINT     CSP09260
                *                 CALL EDIT(JCARD,J,JLAST,KCARD,K,KLAST) CSP09270
                *                      THE WORDS JCARD(J) THROUGH      CSP09280
                *                      JCARD(JLAST) ARE EDITED UNDER   CSP09290
                *                      CONTROL OF THE MASK AT WORDS    CSP09300
                *                      KCARD(K) THROUGH KCARD(KLAST)   CSP09310
                *                      AND THE RESULT IS AT KCARD(K)   CSP09320
                *                      THROUGH KCARD(KLAST).           CSP09330
0000 0  0000    EDIT  DC      *-*  ARGUMENT ADDRESS COMES IN HERE      CSP09340
0001 0  696D          STX   1 SAVE1+1 SAVE IR1                         CSP09350
0002 0  6A6E          STX   2 SAVE2+1 SAVE IR2                         CSP09360
0003 01 65800000      LDX   I1 EDIT PUT ARGUMENT ADDRESS IN IR1        CSP09370
0005 0  C100          LD    1 0    GET JCARD ADDRESS                   CSP09380
0006 0  D02B          STO     JCRD1 SAVE JCARD ADDRESS FOR NZONE       CSP09390
0007 0  D07C          STO     JCRD2 SAVE JCARD ADDRESS FOR NZONE       CSP09400
0008 00 95800002      S     I1 2    SUBTRACT JLAST VALUE               CSP09410
000A 0  8007          A       ONE+1 ADD CONSTANT OF ONE                CSP09420
000B 0  D050          STO     JCARD+1 CREATE JCARD(JLAST) ADDRESS      CSP09430
000C 0  C102    TWO   LD    1 2    GET JLAST ADDRESS                   CSP09440
000D 0  D025          STO     JLAS1 SAVE JLAST ADDRESS FOR NZONE       CSP09450
000E 0  D076          STO     JLAS2 SAVE JLAST ADDRESS FOR NZONE       CSP09460
000F 00 C5800002      LD    I1 2    GET JLAST VALUE                    CSP09470
0011 00 95800001  ONE S     I1 1    SUBTRACT J VALUE                   CSP09480
0013 0  80FE          A       ONE+1 ADD CONSTANT OF ONE                CSP09490
0014 0  4808          BSC     +    CHECK FIELD WIDTH                   CSP09500
0015 0  C0FC          LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE       CSP09510
0016 0  D026          STO     LDXJ+1 SAVE FIELD WIDTH                  CSP09520
0017 0  C104          LD    1 4    GET K ADDRESS                       CSP09530
0018 0  D076          STO     K1   SAVE K ADDRESS FOR FILL             CSP09540
0019 01 D40000C0      STO   L K2   SAVE K ADDRESS FOR FILL             CSP09550
0018 0  C105          LD    1 5    GET KLAST ADDRESS                   CSP09560
001C 0  D073          STO     KLAS1 SAVE KLAST ADDRESS FOR FILL        CSP09570
001D 0  C103          LD    1 3    GET KCARD ADDRESS                   CSP09580
001E 0  D06F          STO     KCRD1 SAVE KCARD ADDRESS FOR FILL        CSP09590
001F 01 D40000BF      STO   L KCRD2 SAVE KCARD ADDRESS FOR FILL        CSP09600
0021 00 95800005      S     I1 5    SUBTRACT KLAST VALUE               CSP09610
0023 0  80EE          A       ONE+1 ADD CONSTANT OF ONE                CSP09620
0024 0  D01A          STO     KCARD+1 CREATE KCARD(KLAST) ADDRESS      CSP09630
0025 0  D07E          STO     KCRD3+1 CREATE KCARD(KLAST) ADDRESS      CSP09640
0026 00 C5800005      LD    I1 5    GET JLAST VALUE                    CSP09650
0028 00 95800004  FOUR S     I1 4    SUBTRACT J VALUE                   CSP09660
002A 0  80E7          A       ONE+1 ADD CONSTANT OF ONE                CSP09670
0028 0  4808          BSC     +    CHECK FIELD WIDTH                   CSP09680
002C 0  C0E5          LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE       CSP09690
002D 0  D00D          STO     LDXK+1 SAVE FIELD WIDTH                  CSP09700
002E 0  7106          MDX   1 6    MOVE OVER SIX ARGUMENTS             CSP09710
002F 0  6943          STX   1 DONE1+1 CREATE RETURN ADDRESS            CSP09720
                *                    REMOVE AND SAVE THE JCARD ZONE   CSP09730
0030 30 15A56545      CALL    NZONE NZONE TO REMOVE SIGN                 CSP09740
0032 0  0000    JCRD1 DC      *-*  ADDRESS OF JCARD                     CSP09750
0033 0  0000    JLAS1 DC      *-*  ADDRESS OF JLAST                     CSP09760
0034 1  0029          DC      FOUR+1 ADDRESS OF A FOUR                 CSP09770
0035 1  00C9          DC      NSIGN ADDRESS OF OLD SIGN INDICATOR      CSP09780
```

```
                        *                   NOUMP=16448                          CSP09790
                        *                   MONEY=16448                          CSP09800
0036 0  C85C            BLANK   LDD     BLANK LOAD TWO BLANKS                    CSP09810
0037 0  D85C            MONEY   STD     MONEY STORE IN MONEY AND NDUMP           CSP09820
                        *                   NZRSP=0                              CSP09830
0038 0  1810            ZRSP    SRA     16  CLEAR THE ACCUMULATOR                CSP09840
0039 0  0050            NZRSP   STO     NZRSP SET NZRSP EQUAL TO ZERO            CSP09850
                        *                   KNOW=KLAST                           CSP09860
003A 00 65000000    L   LDXK    LDX   L1 *-*  LOAD IR1 WITH KCARD COUNT          CSP09870
                        *                   JNOW=JLAST                           CSP09880
003C 00 66000000    L   LDXJ    LDX   L2 *-*  LOAD IR2 WITH JCARD COUNT          CSP09890
                        *                   KTEST=KCARD(KNOW)                    CSP09900
003E 00 C4000000    L   KCARD   LD    L *-*  PICKUP KCARD(KNOW)                  CSP09910
0040 0  00FA            STO     LOXK+1 AND SAVE IT TEMPORARILY                   CSP09920
                        *                   IS KTEST NEGATIVE                    CSP09930
0041 01 4C100047    L       BSC   L POSZ,-  IS IT NEGATIVE-NO-GO TO POSZ         CSP09940
                        *                   IS KTEST EQUAL TO AN EBCDIC ZERO     CSP09950
0043 0  9052            POSZ    S       ZERO YES-CHECK AGAINST EBCDIC ZERO       CSP09960
0044 01 4C20007E        L       BSC   L NEXT,Z IF NOT EQUAL-GO TO NEXT           CSP09970
0046 0  700F            ZRSP    MDX     ZRSP IF EQUAL-GO TO ZRSP                 CSP09980
                        *                   IS KTEST EQUAL TO 16448              CSP09990
0047 0  9048            POSZ    S       BLANK NOT NEGATIVE-CHECK AGAINST EBCD    CSP10000
0048 01 4C180057        L       BSC   L SRCE,+- BLANK-EQUAL-GO TO SRCE           CSP10010
004A 0  C0F0            LOXK    LD      LDXK+1 NOT EQUAL-PICKUP KTEST            CSP10020
                        *                   IS KTEST EQUAL TO 23616              CSP10030
004B 0  9040            DLRSG   S       DLRSG IS IT A DOLLAR SIGN                CSP10040
004C 01 4C180054        L       BSC   L MNY,+- YES-GO TO MNY                     CSP10050
004E 0  C0EC            LOXK    LD      LDXK+1 NO-PICKUP KTEST                   CSP10060
                        *                   IS KTEST EQUAL TO 23360              CSP10070
004F 0  9048            AST     S       AST  IS IT AN ASTERISK                   CSP10080
0050 0  4820            Z       BSC     Z    YES-SKIP NEXT INSTRUCTION           CSP10090
0051 0  702C            NEXT    MDX     NEXT NO-GO TO NEXT                       CSP10100
                        *                   NDUMP=KTEST                          CSP10110
0052 0  C0E8            LOXK    LO      LDXK+1 PICKUP KTEST AND                  CSP10120
0053 0  0041            NDUMP   STO     NDUMP STORE IT IN NDUMP                  CSP10130
                        *                   MONEY=KTEST                          CSP10140
0054 0  C0E6            MNY     LD      LDXK+1 PICKUP KTEST AND                  CSP10150
0055 0  D03E            MONEY   STO     MONEY STORE IT IN MONEY                  CSP10160
                        *                   NZRSP=KNOW                           CSP10170
0056 0  6940            ZRSP    STX   1 NZRSP SAVE KNOW IN NZRSP                 CSP10180
                        *                   SEE IF JNOW IS LESS THAN J.  IF      CSP10190
                        *                   YES, GO TO NEXT. IF NO, GO TO        CSP10200
                        *                   JCARD.                               CSP10210
0057 0  6AA8            SRCE    STX   2 EDIT GET IR1 AND                         CSP10220
0058 0  C0A7            LOXJ    LD      EDIT LOAD ITS VALUE                      CSP10230
0059 01 4C08007E        L       BSC   L NEXT,+ IF NOT POSITIVE-GO TO NEXT        CSP10240
                        *                   KTEST=JCARD(JNOW)                    CSP10250
                        *                   KCARD(KNOW)=KTEST                    CSP10260
0058 00 C4000000    L   JCARD   LD    L *-*  POSITIVE-PICKUP JCARD(JNOW) AND     CSP10270
0050 01 D480003F        STO   I KCARD+1 STORE IT IN KCARD(KNOW)                  CSP10280
005F 0  0000            STO     LDXJ+1 STORE IN KTEST                            CSP10290
                        *                   JNOW=JNOW-1                          CSP10300
0060 0  72FF            MOX   2 -1   DECREMENT IR2                               CSP10310
0061 0  7000            MOX     *    NOP                                         CSP10320
0062 01 7401005C        L       MOX   L JCARD+1,1 MODIFY JCARD ADDRESS TO        CSP10330
```

```
                        *                    JNOW-1                                  CSP10340
                        *                    IS N2RSP POSITIVE                       CSP10350
0064 0  C032                 LD       N2RSP PICKUP N2RSP AND                       CSP10360
0065 01 4C0800TE             BSC  L   NEKT,+ IF NOT POSITIVE-GO TO NEKT            CSP10370
                        *                    IS KTEST NEGATIVE                       CSP10380
006T 0  C0D5                 LD       LDKJ+1 POSITIVE-PICKUP KTEST                 CSP10390
006B 01 4C100074             BSC  L   OVER,- IF NOT NEGATIVE-GO TO OVER            CSP10400
006A 0  902B                 S        ZERO NEGATIVE-CHECK AGAINST ZERO             CSP10410
006B 01 4C1B00TE             BSC  L   NEKT,+- EQUAL-GO TO NEXT                     CSP10420
0060 0  T00D                 MDX      SETAG NOT EQUAL-GO TO SETAG                  CSP10430
                        *                    EXIT.............................       CSP10440
006E 00 65000000     SAVE1   LDK  L1  *-*  RESTORE IR1                            CSP10450
0070 00 66000000     SAVE2   LDK  L2  *-*  RESTORE IR2                            CSP10460
00T2 00 4C000000     00NE1   BSC  L   *-*  RETURN TO CALLING PROGRAM               CSP10470
                        *                    IS KTEST EQUAL TO BLANK                 CSP10480
00T4 0  901E         OVER    S        BLANK CHECK KTEST AGAINST BLANK              CSP10490
0075 01 4C1B007E             BSC  L   NEXT,+- IF EQUAL-GO TO NEKT                  CSP10500
                        *                    IS KTEST EQUAL TO COMMA                 CSP10510
007T 0  C0C5                 LO       LDXJ+1 NOT EQUAL-CHECK KTEST                 CSP10520
007B 0  9021                 S        COMMA AGAINST A COMMA                        CSP10530
00T9 01 4C1B007E             BSC  L   NEXT,+- EQUAL-GO TO NEKT                     CSP10540
                        *                    N2RSP=KNOW-1                            CSP10550
007B 0  691B         SETAG   STX    1 NZRSP NOT EQUAL-SET N2RSP EQUAL TO           CSP10560
007C 01 74FF009T             MDK  L   N2RSP,-1 KCARD COUNT MINUS ONE               CSP10570
                        *                    KNOW=KNOW-1                             CSP10580
                        *                    SEE IF KNOW IS LESS THAN K.  IF         CSP10590
                        *                    YES, PUT JCARD ZONE BACK.  IF NO        CSP10600
                        *                    GO BACK FOR MORE.                       CSP10610
007E 01 7401003F     NEXT    MQX  L   KCARD+1,1 MODIFY KCARD ADDRESS TO             CSP10620
                        *                    KNOW-1                                  CSP10630
00B0 0  71FF                 MQK    1 -1   DECREMENT IR1                           CSP10640
00B1 0  T0BC                 MDK      KCARD GO BACK FOR MORE                       CSP10650
                        *                    PUT JCARD ZONE BACK                     CSP10660
00B2 30 15A56545             CALL     NZONE RESTORE JCARD ZONE                     CSP10670
00B4 0  0000         JCR02   DC       *-*  ADDRESS OF JCARD                        CSP10680
00B5 0  0000         JLAS2   DC       *-*  ADDRESS OF JLAST                        CSP10690
00B6 1  00C9                 DC       NSIGN ADDRESS OF NEW SIGN INDICATOR          CSP10700
00BT 1  0000                 DC       EDIT DUMMY                                   CSP10710
                        *                    SEE IF JNOW IS LESS THAN J.  IF         CSP10720
                        *                    YES, GO TO OK.  IF NO, FILL WITH        CSP10730
                        *                    ASTERISKS AND EXIT                      CSP10740
00BB 0  6AA9                 STK    2 JCR01 GET THE CONTENTS OF                    CSP10750
00B9 0  C0AB                 LD       JCRD1 IR2 AND CHECK                          CSP10760
00BA 01 4C0B009F             BSC  L   OK,+ IF NOT POSITIVE-GO TO OK                CSP10770
008C 30 062534C0             CALL     FILL POSITIVE-ERROR-JCARD TOO LONG           CSP10780
                        *                    FILL KCARD WITH ASTERISKS               CSP10790
00BE 0  0000         KCR01   OC       *-*  ADDRESS OF KCARD                        CSP10800
00BF 0  0000         K1      DC       *-*  ADDRESS OF K                            CSP10810
0090 0  0000         KLAS1   DC       *-*  ADDRESS OF KLAST                        CSP10820
0091 1  0098                 DC       AST  ADDRESS OF FILL CHARACTER               CSP10830
0092 0  70DB                 MDK      SAVE1 GO TO EXIT                             CSP10840
0093 0  4040         BLANK   DC       /4040 CONSTANT OF EBCDIC BLANK               CSP10850
0094 0  0000         MONEY   OC       *-*  FILL FOR FLOATING $                     CSP10860
0095 0  0000         NOUMP   DC       *-*  FILL FOR ANY SUPPRESSION                CSP10870
0096 0  F040         2ERO    DC       /F040 CONSTANT OF EBCDIC ZERO                CSP10880
```



```
009T 0  0000         N2RSP   DC       *-*  HOW FAR TO ZERO SUPPRESS                CSP10890
009B 0  5C40         AST     OC       /5C40 CONSTANT OF ASTERISK                   CSP10900
0099 0  5B40         DLRSG   OC       /5B40 CONSTANT OF DOLLAR SIGN                CSP10910
009A 0  6B40         COMMA   DC       /6B40 CONSTANT OF COMMA                      CSP10920
009B 0  6040         MINUS   DC       /6040 CONSTANT OF MINUS SIGN                 CSP10930
009C 0  0940         R       DC       /0940 CONSTANT OF LETTER R                   CSP10940
009D 0  0001         ONE2    DC       1    CONSTANT OF ONE                         CSP10950
009E 0  0002         TWO2    OC       2    CONSTANT OF TWO                         CSP10960
                        *                    IS NSIGN EQUAL TO TWO                   CSP10970
009F 0  C029         OK      LO       NSIGN PICKUP THE ORIGINAL ZONE               CSP10980
00A0 0  90F0                 S        TWO2 INDICATOR AND CHECK AGAINST TWO         CSP10990
00A1 01 4C1B00B6             BSC  L   NEG,+- EQUAL-GO TO NEG                       CSP11000
                        *                    KTEST=KCARD(KLAST)                      CSP11010
00A3 00 C4000000     KCR03   LO   L   *-*  NOT EQUAL-PICKUP KCAR0(KLAST)           CSP11020
00A5 0  90F5                 S        MINUS AND CHECK AGAINST MINUS SIGN           CSP11030
00A6 01 4C1B00B3             BSC  L   LD2,+- IF EQUAL-GO TO LO2                    CSP11040
00AB 0  B0F2                 A        MINUS NOT EQUAL-GET KTEST AND CHECK          CSP11050
00A9 0  90F2                 S        R    AGAINST LETTER R                        CSP11060
00AA 01 4C2000B6             BSC  L   NEG,Z IF NOT EQUAL-GO TO NEG                 CSP11070
00AC 01 740100A4             MDX  L   KCR03+1,1 EQUAL-GET ADDRESS OF               CSP11080
                        *                    KCAR0(KLAST-1)                          CSP11090
                        *                    KCARD(KLAST-1)=1644B                    CSP11100
00AE 0  C0E4                 LO       BLANK PICKUP A BLANK                         CSP11110
00AF 01 D4B000A4             STO  I   KCR03+1 STORE AT KCARD(KLAST-1)              CSP11120
00B1 01 74FF00A4             MDK  L   KCR03+1,-1 GET ADDR OF KCAR0(KLAST)          CSP11130
                        *                    KCARD(KLAST)=1644B                      CSP11140
00B3 0  C0DF         LD2     LD       BLANK PICKUP A BLANK                         CSP11150
00B4 01 D4B000A4             STO  I   KCR03+1 STORE AT KCAR0(KLAST)               CSP11160
                        *                    IS N2RSP GREATER THAN ZERO              CSP11170
00B6 0  C0E0         NEG     LD       NZRSP GET NZRSP AND                          CSP11180
00BT 01 4C08006E             BSC  L   SAVE1,+ IF NOT POSITIVE-EXIT                 CSP11190
00B9 01 B4B000BF             A    I   K1   POSITIVE-CALCULATE SUBSCRIPT OF        CSP11200
00BB 0  90E1                 S        ONE2 LAST POSITION TO BE ZERO                CSP11210
00BC 0  D0E7                 STO      KCRD3+1 SUPPRESSED-END OF FILL AREA          CSP11220
                        *                    ZERO SUPPRESS                           CSP11230
00BD 30 062534C0             CALL     FILL FILL ROUTINE TO ZERO SUPPRESS           CSP11240
00BF 0  0000         KCRD2   DC       *-*  ADDRESS OF KCARD                        CSP11250
00C0 0  0000         K2      DC       *-*  ADDRESS OF K                            CSP11260
00C1 1  00A4                 DC       KCR03+1 ADDRESS OF END OF FILL AREA          CSP11270
00C2 1  0095                 DC       NDUMP ADDRESS OF FILL CHARACTER              CSP11280
                        *                    KCARD(NZRSP)=MONEY                      CSP11290
00C3 0  C0FB                 LO       KCR02 GET KCARD ADDRESS                      CSP11300
00C4 0  90DF                 S        KCRD3+1 SUBTRACT LAST FILL VALUE             CSP11310
00C5 0  B0DT                 A        ONE2 ADD CONSTANT OF ONE                     CSP11320
00C6 0  D002                 STO      STOK+1 CREATE KCAR0(NZRSP) ADDRESS           CSP11330
00CT 0  C0CC                 LD       MONEY PICKUP MONEY VALUE                     CSP11340
00CB 00 D4000000     STOK    STO  L   *-*  STORE FOR SUPPRESSION                   CSP11350
00C9                 NSIGN   EQU      STOK+1 TO SAVE CORE STORAGE                  CSP11360
00CA 0  70A3                 MDX      SAVE1 GO TO EXIT                             CSP11370
00CC                         END                                                   CSP11380
```

NO ERRORS IN ABOVE ASSEMBLY.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

```
// DUP                                                                  CSP11390
*STORE      WS  UA  EDIT                                                 CSP114DD
3361 00DD
```

```
// ASM                                                                  CSP1141D
** FILL SUBROUTINE FOR 113D COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP1142D
* NAME FILL                                                        (ID) CSP1143D
* LIST                                                                 CSP1144D
0000     062534C0           ENT     FILL FILL SUBROUTINE ENTRY PDINT     CSP11450
                       *                  CALL FILL(JCARD,J,JLAST,NCH)     CSP11460
                       *                  THE WORDS JCARD(J) THROUGH       CSP11470
                       *                  JCARD(JLAST) ARE FILLED WITH THE CSP1148D
                       *                  CHARACTER AT LOCATIDN NCH.       CSP11490
0000 0   00C0         FILL    DC      *-*  ARGUMENT ADDRESS COMES IN HERE   CSP11500
0001 0   6919                STX   1 SAVE1+1 SAVE IR1                     CSP1151D
0002 01  6580D000            LDX   I1 FILL PUT ARGUMENT ADDRESS IN IR1    CSP11520
0004 0   C100                LD    1 0    GET JCARD ADDRESS               CSP11530
0005 00  95800002            S     I1 2    SUBTRACT VALUE DF JLAST         CSP11540
0007 0   D00F                STO     STO+1 CREATE ADDRESS OF JCARD(JLAST)  CSP11550
0008 00  C5800D02            LD    I1 2    GET VALUE DF JLAST              CSP1156D
000A 00  95800D01    ONE     S     I1 1    SUBTRACT VALUE OF J             CSP11570
000C 0   80FE                A       ONE+1 ADD CONSTANT OF DNE             CSP1158D
000D 0   4808                BSC     +     CHECK FIELD WIDTH               CSP1159D
000E 0   C0FC                LD      DNE+1 NEGATIVE DR ZERD - MAKE IT ONE  CSP116D0
00DF 0   D005                STO     LDX+1 OK - STORE FIELD WIDTH IN LDX   CSP1161D
0010 00  C5800D03            LD    I1 3    GET FILL CHARACTER - NCH        CSP11620
0012 0   7104                MDX   1 4    MOVE OVER FOUR ARGUMENTS         CSP1163D
0013 0   6909                STX   1 DONE1+1 CREATE RETURN ADDRESS         CSP11640
                       *                  JNOW=J                           CSP11650
0014 00  65000000    LDX     LDX   L1 *-*  LOAD IR1 WITH FIELD WIDTH        CSP1166D
                       *                  JCARD(JNOW)=NCH                  CSP1167D
0016 00  D5000D00    STO     STO   L1 *-*  STORE FILL CHAR AT JCARD(JNOW)   CSP1168D
                       *                  SEE IF JNOW IS LESS THAN JLAST.  CSP1169D
                       *                  IF YES, JNOW=JNOW+1 AND GO BACK  CSP117D0
                       *                  FOR MORE.  IF NO, EXIT.          CSP1171D
0018 0   71FF                MDX   1 -1   DECREMENT FIELD WIDTH            CSP11720
0019 0   7DFC                MOX     STO  NDT DONE - GO BACK FDR MORE      CSP1173D
                       *                  EXIT.........                    CSP1174D
001A 00  6500D000    SAVE1   LDX   L1 *-*  DONE - RESTORE IR1               CSP1175D
001C 00  4C00DD00    DONE1   BSC   L  *-*  RETURN TO CALLING PRDGRAM         CSP1176D
001E                 FILL    END                                           CSP1177D
```

NO ERRORS IN ABDVE ASSEMBLY.

```
// DUP                                                                  CSP1178D
*STORE      WS  UA  FILL                                                 CSP1179D
336E 0DD3
```

```
// ASM                                                                  CSP11800
** GET SUBRDUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP11810
* NAME GET                                                        (ID) CSP11820
* LIST                                                                CSP11830
0000    07163000          ENT     GET  GET SUBRDUTINE ENTRY POINT     CSP11840
                     *                GET(JCARD,J,JLAST,SHIFT)        CSP11850
                     *                THE WORDS JCARD(J) THROUGH      CSP11860
                     *                JCARD(JLAST) ARE CONVERTED TO A CSP11870
                     *                REAL NUMBER AND MULTIPLIED BY   CSP11880
                     *                SHIFT TO PLACE THE DECIMAL POINT CSP11890
0000 0  0000      GET   DC      *-*  ARGUMENT ADDRESS COMES IN HERE    CSP11900
0001 0  694B            STX   1 FIN+1 SAVE IR1                        CSP11910
0002 01 65800000        LDX   I1 GET  PUT ARGUMENT ADDRESS IN IR1     CSP11920
0004 0  C100            LD    1 0     GET JCARD ADDRESS               CSP11930
0005 0  D013            STO     JCRD1 STORE FOR NZONE AT JCRD1        CSP11940
0006 0  D03C            STD     JCRD3 STORE FOR N2ONE AT JCRD3        CSP11950
0007 00 95800002  TWO   S     I1 2    SUBTRACT JLAST VALUE            CSP11960
0009 0  D01B            STO     JCRD2+1 CREATE JCARD(JLAST) ADDRESS   CSP11970
000A 0  C103            LD    1 3     GET SHIFT ADDRESS AND           CSP11980
000B 0  D033            STO     SHIFT STORE FOR MULTIPLY TO PLACE  .  CSP11990
000C 00 C5800002        LD    I1 2    GET JLAST VALUE AND             CSP12000
000E 0  D0F1            STO     GET   SAVE FOR NZONE                  CSP12010
000F 00 95800001  ONE   S     I1 1    SUBTRACT J VALUE                CSP12020
0011 0  80FE            A       ONE+1 ADD CONSTANT OF ONE             CSP12030
0012 0  4808            BSC     +     CHECK FIELD WIDTH               CSP12040
0013 0  C0FC            LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE    CSP12050
0014 0  D00E            STD     CNT+1 OK-SAVE FIELD WIDTH AT COUNT    CSP12060
0015 0  7104            MDX   1 4     MOVE OVER FOUR ARGUMENTS        CSP12070
0016 0  6938            STX   1 DONE1+1 CREATE RETURN ADDRESS         CSP12080
                     *                MAKE THE FIELD POSITIVE AND     CSP12090
                     *                SAVE THE ORIGINAL SIGN          CSP12100
0017 30 15A56545        CALL    N2ONE NZONE TO CLEAR ORIGINAL SIGN    CSP12110
0019 0  0000      JCRD1 DC      *-*  ADDRESS OF JCARD                 CSP12120
001A 1  0000            DC      GET  ADDRESS OF JLAST                 CSP12130
001B 1  0050            DC      FOUR AODRESS OF CONSTANT OF FOUR      CSP12140
001C 1  0019            DC      JCRD1 ADORESS OF OLD SIGN INDICATOR   CSP12150
001D 0  1BA0            SRT     32   CLEAR ACCUMULATOR AND EXTENSION  CSP12160
001E 0  DB7E            STD   3 126  CLEAR MANTISSA OF FAC            CSP12170
001F 0  D37D            STO   3 125  CLEAR CHARACTERISTIC OF FAC      CSP12180
                     *                LET GET AND ANS BE EQUIVALENT   CSP12190
0020 20 058A3580        LIBF    ESTO STORE THE CONTENTS OF FAC        CSP12200
0021 1  005A            DC      ANS  AT GET                           CSP12210
                     *                JNOW=J                          CSP12220
0022 00 65000000  CNT   LDX   L1 *-* LOAD IR1 WITH THE FIELD WIDTH    CSP12230
                     *                JTEST=JCARD(JNOW)               CSP12240
0024 00 C5000000  JCRD2 LD    L1 *-* PICKUP JCARD(JNOW)               CSP12250
0026 01 4C28002C        BSC   L  MAYBE,+Z IS JTEST NEGATIVE-YES-MAYBE CSP12260
0028 0  9028            S       BLANK NO - IS JTEST EQUAL TO AN       CSP12270
0029 01 4C200053        BSC   L  ERR,2 EBCDIC BLANK - NO - GO TO ERR  CSP12280
002B 0  C026            LD      ZERO YES - REPLACE BLANK WITH ZERO    CSP12290
002C 0  9025      MAYBE S       ZERO IS JTEST LESS THAN AN EBCDIC     CSP12300
002D 01 4C280053        BSC   L  ERR,+2 ZERO - YES - GO TO ERR        CSP12310
                     *                JTEST+4032 IN ACCUMULATOR       CSP12320
                     *                GET=10*GET+(JTEST+4032)/256     CSP12330
                     *                SHIFT 8 IS SAME AS DIVIDE BY 256 CSP12340
002F 0  1B08            SRA     8    NO - SHIFT 4 BIT DIGIT TO LOW    CSP12350
0030 20 064D6063        LIBF    FLOAT ORDER OF ACC AND MAKE REAL      CSP12360
```

```
                                                            PAGE  2
0031 20 05BA3580        LIBF    ESTO STORE REAL DIGIT                 CSP12370
0032 1  0057            DC      TEMP IN TEMPORARY STORAGE             CSP12380
0033 20 054C4000        LIBF    ELO  LOAD FAC WITH                    CSP12390
0034 1  005A            DC      ANS  GET                              CSP12400
0035 20 05517A00        LIBF    EMPY MULTIPLY GET                     CSP12410
0036 1  005D            DC      ETEN BY TEN                           CSP12420
0037 20 15599500        LIBF    NORM NORMALIZE THE PRODUCT            CSP12430
0038 20 05044100        LIBF    EADD ADD TEMPORARY STORAGE            CSP12440
0039 1  0057            DC      TEMP TO FAC                           CSP12450
003A 20 05BA3580        LIBF    ESTO STORE RESULT                     CSP12460
003B 1  005A            DC      ANS  IN GET                           CSP12470
                     *                SEE IF JNOW IS LESS THAN JLAST.  CSP12480
                     *                IF YES, JNOW=JNOW+1 AND GO BACK CSP12490
                     *                FOR MORE.  IF NO, PLACE DECIMAL CSP12500
                     *                POINT.                          CSP12510
003C 0  71FF            MDX   1 -1   DECREMENT FIELD WIDTH            CSP12520
003D 0  70E6            MDX     JCRD2 NOT DONE-GET NEXT DIGIT         CSP12530
                     *                GET=SHIFT*GET                   CSP12540
003E 20 05517A00        LIBF    EMPY DONE-MULTIPLY BY SHIFT TO PLACE  CSP12550
003F 0  0000      SHIFT DC      *-*  ADDRESS OF SHIFT---DECIMAL POINT CSP12560
0040 20 15599500        LIBF    NORM NORMALIZE THE RESULT             CSP12570
                     *                REPLACE SIGN OF JCARD           CSP12580
0041 30 15A56545        CALL    NZONE RESTORE ORIGINAL JCARD SIGN     CSP12590
0043 0  0000      JCRD3 DC      *-*  ADDRESS OF JCARD                 CSP12600
0044 1  0000            DC      GET  ADDRESS OF JLAST                 CSP12610
0045 1  0019            DC      JCRD1 ADDRESS OF ORIG. SIGN INDICATOR CSP12620
0046 1  0043            DC      JCRD3 DUMMY                           CSP12630
                     *                IF INDICATOR EQUALS 2,          CSP12640
                     *                GET=-GET.  OTHERWISE, EXIT..... CSP12650
0047 0  C0D1            LD      JCRD1 LOAD OLD SIGN AND SEE IF IT     CSP12660
004A 0  90BF            S       TWO+1 WAS NEGATIVE                    CSP12670
0049 01 4C20004C        BSC   L  FIN,2 IF YES,REVERSE SIGN-NO-EXIT    CSP12680
                     *                GET=-GET                        CSP12690
004B 20 22559000        LIBF    SNR  REVERSE THE SIGN OF THE RESULT   CSP12700
                     *                EXIT.........................   CSP12710
004C 00 65000000  FIN   LDX   L1 *-* RESTORE IR1                      CSP12720
004E 00 4C000000  DONE1 BSC   L  *-* RETURN TO CALLING PROGRAM        CSP12730
0050 0  0004      FOUR  DC      4    CONSTANT OF FOUR                 CSP12740
0051 0  4040      BLANK DC      /4040 CONSTANT OF EBCDIC BLANK        CSP12750
0052 0  F040      ZERO  DC      /F040 CONSTANT OF EBCDIC ZERO         CSP12760
0053 0  10A0      ERR   SLT     32   CLEAR ACCUMULATOR AND EXTENSION  CSP12770
0054 0  DB7E            STD   3 126  CLEAR MANTISSA OF FAC            CSP12780
0055 0  D37D            STO   3 125  CLEAR CHARACTERISTIC OF FAC      CSP12790
0056 0  70F5            MDX     FIN  GO TO EXIT                       CSP12800
0057    0003      TEMP  BSS     3    TEMPORARY STORAGE                CSP12810
005A    0003      ANS   BSS     3    TEMPORARY STORAGE                CSP12820
005D 84 50000000  ETEN  XFLC    10.0 CONSTANT OF 10.0 (TEN)           CSP12830
0060              END                                                 CSP12840
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP1285D
*STDRE      WS  UA  GET                                                  CSP12860
3371 00D7
```

```
// ASM                                                                  CSP12870
** ICOMP SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP12880
* NAME ICOMP                                                        (IO) CSP12890
* LIST                                                                 CSP12900
0000    D90D6517          ENT     ICOMP ICOMP SUBRDUTINE ENTRY POINT   CSP12910
                      *               ICOMP(JCARD,J,JLAST,KCARD,K,KLAST) CSP12920
                      *                  THE WORDS JCARO(J) THROUGH     CSP12930
                      *                  JCARO(JLAST) ARE COMPARED TO THE CSP12940
                      *                  WORDS KCARD(K) THROUGH         CSP12950
                      *                  KCARO(KLAST).                  CSP12960
0000 0  0000        ICOMP DC      *-* ARGUMENT AODRESS COMES IN HERE     CSP12970
0001 0  6972              STX     1 SAVE1+1 SAVE IR1                     CSP12980
0002 01 65800000          LDX   I1 ICOMP PUT ARGUMENT ADDRESS IN IR1     CSP12990
0004 0  C100              LO      1 0     GET JCARO AODRESS              CSP13000
0005 00 95800002          S     I1 2      SUBTRACT JLAST VALUE           CSP13010
0007 0  D048              STO       JPIC1+1 STORE JCARD(JLAST) FOR JHASH CSP13020
0008 0  004A              STO       JPIC2+1 STORE JCARO(JLAST) FOR ICOMP CSP13030
0009 0  800A              A         ONE+1 ADD CDNSTANT DF ONE            CSP13040
000A 0  000F              STO       SGNJ+1 CREATE AODRESS OF JCARD(JLAST) CSP13050
000B 0  C103              LO      1 3     GET KCARD ADDRESS              CSP13060
000C 00 95800005          S     I1 5      SU8TRACT KLAST VALUE           CSP13070
000E 0  D046              STO       KPIC2+1 STDRE KCARD(KLAST) FOR ICOMP CSP13080
000F 0  8004              A         ONE+1 ADD CONSTANT OF ONE            CSP13090
0010 0  0011              STO       SGNK+1 CREATE ADDRESS OF KCARO(KLAST) CSP13100
0011 00 C5800002    TWO   LD      I1 2      GET VALUE OF JLAST           CSP13110
0013 00 95800001    ONE   S     I1 1      SU8TRACT VALUE OF J          CSP13120
0015 0  80FE              A         ONE+1 ADD CONSTANT OF ONE            CSP13130
0016 0  4808              8SC       +     CHECK FIELD WIDTH              CSP13140
0017 0  C0FC              LD        ONE+1 NEGATIVE DR ZERO-MAKE IT ONE   CSP13150
0018 0  D035              STO       CNTCO+1 SAVE FIELD WIDTH IN CDMP CNT CSP13160
                      *                 CLEAR AND SAVE THE SIGNS ON THE  CSP13170
                      *                 JCARO AND THE KCARD FIELOS       CSP13180
0019 00 C4000000    SGNJ  LD    L *-* PICKUP THE SIGN DF JCARO           CSP13190
0018 0  D058              STD       JSIGN SAVE IT                        CSP13200
001C 01 4C100021          BSC   L SGNK,- IS IT NEG-ND-LDOK AT KCARD      CSP13210
001E 0  F00F              EOR       HFFFF+1 YES-MAKE IT PDSITIVE AND     CSP13220
001F 01 D480001A          STD   I SGNJ+1 CHANGE JCARD FIELD SIGN         CSP13230
0021 00 C4000000    SGNK  LD    L *-* PICKUP THE SIGN DF KCARD           CSP13240
0023 0  0054              STO       KSIGN SAVE IT                        CSP13250
0024 01 4C100029          8SC   L CHCK,- IS IT NEG-NO-GD TO CHCK       CSP13260
0026 0  F007              EOR       HFFFF+1 YES-MAKE IT PDSITIVE AND     CSP13270
0027 01 D4800022          STO   I SGNK+1 CHANGE THE KCARD FIELD SIGN     CSP13280
0029 0  7106        CHCK  MDX     1 6     MOVE OVER SIX ARGUMENTS        CSP13290
002A 0  6948              STX     1 DDNE1+1 CREATE RETURN AODRESS        CSP13300
                      *                 K IS COMPARED TO                 CSP13310
                      *                 KSTRT=KLAST+J-JLAST-1            CSP13320
0028 00 C580FFFE          LD    I1 -2     PICKUP THE VALUE DF K          CSP13330
002D 00 9580FFFF    HFFFF S     I1 -1     SU8TRACT THE VALUE OF KLAST    CSP13340
002F 00 9580FFF8          S     I1 -5     SU8TRACT THE VALUE OF J        CSP13350
0031 00 8580FFFC          A     I1 -4     ADD THE VALUE DF JLAST         CSP13360
0033 0  80E0              A         ONE+1 ADD CONSTANT OF DNE            CSP13370
0034 01 4C300048          8SC   L JHASH,-2 IF POSITIVE GD TO JHASH       CSP13380
0036 0  F0F7              EOR       HFFFF+1 OTHERWISE COMPLIMENT AND ADD  CSP13390
0037 0  80DA              A         TWO+1 ONE GIVING LEADING PART KCARO  CSP13400
0038 0  D008              STD       21PCT+1 STORE THIS CDUNT AT 21PCT    CSP13410
0039 00 8580FFFE          A     I1 -2     ADD VALUE OF K                 CSP13420
```

```
0038 0  90D8              S       ONE+1 SUBTRACT CONSTANT OF ONE          CSP13430
003C 0  D0C3              STO     ICOMP STORE TEMPORARILY                 CSP13440
003D 0  C1FD              LD    1 -3   GET KCARD ADDRESS                 CSP13450
003E 0  90C1              S       ICOMP SUBTRACT TEMPORARY VALUE GIVING   CSP13460
003F 0  D006              STO     KPIC1+1 ADDR FOR SEARCHING BEGINNING    CSP13470
                    *                 OF KCARD                          CSP13480
                    *                 ICOMP=-KSIGN                      CSP13490
0040 0  C037              LD      KSIGN LOAD SIGN OF KCARD                CSP13500
0041 0  F0EC              EOR     HFFFF+1 NEGATE IT                       CSP13510
0042 0  D0BD              STO     ICOMP STORE IT IN ICOMP                 CSP13520
                    *                 KNOW=K                            CSP13530
0043 00 65000000    ZIPCT LDX   L1 *-*  LOAD IR1 WITH BEGINNING KCARD CT CSP13540
0045 00 C5000000    KPIC1 LD    L1 *-*  PICKUP KCARD(KNOW)               CSP13550
                    *                 IS KCARD(KNOW) POSITIVE           CSP13560
0047 01 4C30006C          BSC   L FIN,-Z IF POSITIVE, GO TO FIN         CSP13570
                    *                 SEE IF KNOW IS LESS THAN KSTRT.   CSP13580
                    *                 IF YES, KNOW=KNOW+1 AND LOOK AT   CSP13590
                    *                 NEXT KCARD WORD.  IF NO, GO TO    CSP13600
                    *                 JHASH.                            CSP13610
0049 0  71FF              MDX     1 -1    OTHERWISE, DECREMENT FIELD WIDTH CSP13620
004A 0  70FA              MDX     KPIC1 NOT DONE=GO BACK FOR NEXT DIGIT CSP13630
                    *                 JHASH=0                           CSP13640
004B 0  1810        JHASH SRA       16    DONE=CLEAR ACCUMULATOR        CSP13650
004C 0  D083              STO     ICOMP CLEAR ICOMP                     CSP13660
                    *                 KNOW=KSTRT+1                      CSP13670
                    *                 KSTRT=J                           CSP13680
004D 00 65000000    CNTCO LDX   L1 *-*  LOAD IR1 WITH FIELD WIDTH       CSP13690
                    *                 JHASH=JHASH+JCARD(KSTRT)          CSP13700
004F 00 85000000    JPIC1 A     L1 *-*  ADD JCARD(KSTRT) TO JHASH        CSP13710
0051 0  1B90              SRT     16    STORE JHASH IN EXTENSION        CSP13720
                    *                 ICOMP=JCARD(KSTRT)-KCARD(KNOW)    CSP13730
0052 00 C5000000    JPIC2 LD    L1 *-*  LOAD JCARD(KSTRT)                CSP13740
0054 00 95000000    KPIC2 S     L1 *-*  SUBTRACT KCARD(KNOW)             CSP13750
0056 0  D0A9              STO     ICOMP STORE RESULT                    CSP13760
                    *                 IS ICOMP ZERO - NO - GO TO NEO    CSP13770
0057 01 4C200063          BSC   L NEO,Z IF NOT ZERO, GO TO NEO.         CSP13780
0059 0  1090              SLT     16    OTHERWISE, PUT JHASH IN ACCUM   CSP13790
                    *                 KNOW=KNOW+1                       CSP13800
                    *                 SEE IF KSTRT IS LESS THAN JLAST.  CSP13810
                    *                 IF YES, KSTRT=KSTRT+1 AND TRY     CSP13820
                    *                 NEXT PAIR OF DIGITS.  IF NO,      CSP13830
005A 0  71FF              MDX     1 -1    DECREMENT FIELD WIDTH         CSP13840
005B 0  70F3              MDX     JPIC1 NOT OONE - GO BACK              CSP13850
                    *               IF NO IS JSIGN*KSIGN*JHASH NEGATIVE. CSP13860
005C 01 4C18006C          BSC   L FIN,+- DONE=IF JHASH IS ZERO GO FIN   CSP13870
005E 0  C01B              LD      JSIGN OTHERWISE - COMPUTE JSIGN       CSP13880
005F 0  F018              EOR     KSIGN TIMES KSIGN                     CSP13890
0060 01 4C10006C          BSC   L FIN,- IF NOT NEGATIVE, GO TO FIN      CSP13900
0062 0  7004              MDX     OVR1 OTHERWISE GO TO OVR1             CSP13910
                    *                 IS KSIGN*JSIGN NEGATIVE           CSP13920
0063 0  C013        NEG   LD        JSIGN COMPUTE JSIGN                 CSP13930
0064 0  F013              EOR     KSIGN TIMES KSIGN                     CSP13940
0065 01 4C100069          BSC   L OVR2,- IF NOT NEGATIVE, GO TO OVR2    CSP13950
                    *                 ICOMP=1                           CSP13960
0067 0  C0E5        OVR1  LD        CNTCO OTHERWISE, SET ICOMP          CSP13970
```



```
0068 0  D097              STO     ICOMP TO A POSITIVE NUMBER            CSP13980
                    *                 ICOMP=JSIGN*ICOMP                 CSP13990
0069 0  C096        OVR2  LD        ICOMP LOAD ICOMP AND                CSP14000
006A 0  F00C              EOR     JSIGN MULTIPLY BY JSIGN               CSP14010
0068 0  D094              STO     ICOMP STORING THE RESULT IN ICOMP     CSP14020
                    *                 RESTORE THE SIGNS ON THE JCARD    CSP14030
                    *                 AND THE KCARD FIELDS              CSP14040
006C 0  C00A        FIN   LD        JSIGN RESTORE THE ORIGINAL          CSP14050
006D 01 D4B0001A          STO   I SGNJ+1 SIGN OF JCARD                  CSP14060
006F 0  C008              LD      KSIGN RESTORE THE ORIGINAL            CSP14070
0070 01 D4800022          STO   I SGNK+1 SIGN OF KCARD                  CSP14080
0072 0  C08D              LD      ICOMP PUT ICOMP IN THE ACCUMULATOR    CSP14090
                    *                 EXIT                              CSP14100
0073 00 65000000    SAVE1 LDX   L1 *-*  RESTORE IR1                     CSP14110
0075 00 4C000000    DONE1 BSC   L *-*  RETURN TO CALLING PROGRAM         CSP14120
0077 0  0000        JSIGN DC        *-*  SIGN OF JCARD                  CSP14130
0078 0  0000        KSIGN DC        *-*  SIGN OF KCARD                  CSP14140
007A                      END                                           CSP14150
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP14160
*STORE      WS  UA  ICOMP                                               CSP14170
3378 0008
```

```
// ASM                                                                  CSP14180
** IONO SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP14190
* NAME IONO                                                        (ID) CSP14200
* LIST                                                                 CSP14210
0000     09595100           ENT     IOND       SUBROUTINE NAME        CSP14220
                    *CALL IOND     NO PARAMETERS                       CSP14230
                    *CALL IONO     ALLOWS I/O OPERATIONS TO ENO BEFORE A CSP14240
                    *              PAUSE OR STOP IS ENTERED            CSP14250
0000     0001       IONO  BSS      1          ARGUMENT AOORESS         CSP14260
0001 00 74000032    IOPND MDX  L   50,0       ANY INTERRUPTS PENOING   CSP14270
0003 0  70FD          MOX     IOPNO      YES - KEEP CHECKING           CSP14280
0004 01 4C800000    BACK  BSC  I   IONO       NO - RETURN TO CALLING PRG CSP14290
0006                      END                                          CSP14300

    NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                                  CSP14310

*STORE      WS  UA  IOND                                               CSP14320

3380 0002
```

```
// ASM                                                                  CSP14330
** MOVE SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP14340
* NAME MOVE                                                        (ID) CSP14350
* LIST                                                                 CSP14360
0000     145A5140           ENT     MOVE MOVE SUBROUTINE ENTRY POINT     CSP14370
                    *                   CALL MOVE(JCARO,J,JLAST,KCARO,K) CSP14380
                    *                   THE WORDS JCARD(J) THROUGH       CSP14390
                    *                   JCARD(JLAST) ARE MOVEO TO KCARD  CSP14400
                    *                   STARTING AT KCARD(K).            CSP14410
0000 0  0000        MOVE  0C       *-*  ARGUMENT ADDRESS COMES IN HERE   CSP14420
0001 0  691F          STX   1 SAVE1+1 SAVE IR1                           CSP14430
0002 01 65800000      LDX   I1 MOVE  PUT ARGUMENT AODRESS IN IR1         CSP14440
0004 0  C100          LD    1 0      GET JCARD ADDRESS                   CSP14450
0005 00 95800002      S     I1 2     SUBTRACT JLAST VALUE                CSP14460
0007 0  0013          STO      LD1+1 PLACE AODR OF JCARD(JLAST) IN       CSP14470
                    *                   PICKUP OF MOVE                   CSP14480
0008 00 C5800002      LO    I1 2     GET JLAST VALUE                     CSP14490
000A 00 95800001    ONE   S     I1 1     SUBTRACT J VALUE              CSP14500
000C 0  4B28          8SC      +Z       CHECK FIELD WIDTH               CSP14510
000D 0  1810          SRA      16       NEGATIVE - MAKE IT ZERO         CSP14520
000E 0  D00A          STO      LDX+1 STORE FIELD WIDTH IN LOX          CSP14530
000F 0  C103          LO    1 3      GET KCARD ADORESS                  CSP14540
0010 00 95800004      S     I1 4     SUBTRACT K VALUE                   CSP14550
0012 0  9006          S        LDX+1 SUBTRACT FIELD WIDTH               CSP14560
0013 0  D009          STO      STO+1 PLACE ADDR OF KCARD(KLAST) IN      CSP14570
                    *                   STORE OF MOVE                    CSP14580
0014 01 74010019      MOX   L  LDX+1,1       ADD ONE TO FIELD WIOTH      CSP14590
                    *                   MAKING IT TRUE                   CSP14600
0016 0  7105          MDX   1 5      MOVE OVER FIVE ARGUMENTS            CSP14610
0017 0  690B          STX   1 DONE1+1 CREATE RETURN ADDRESS              CSP14620
                    *                   JNOW=J                           CSP14630
                    *                   KNOW=K+JNOW-J                    CSP14640
0018 00 65000000    LDX   LDX  L1 *-*  LOAO IR1 WITH FIELD WIOTH         CSP14650
                    *                   KCARO(KNOW)=JCARD(JNOW)          CSP14660
001A 00 C5000000    LD1   LD   L1 *-*  PICKUP JCARO(JNOW)                CSP14670
001C 00 05000000    STO   STO  L1 *-*  STORE IT IN KCARO(KNOW)           CSP14680
                    *                   SEE IF JNOW IS LESS THAN JLAST.  CSP14690
                    *                   IF YES, JNOW=JNOW+1 AND MOVE     CSP14700
                    *                   NEXT CHARACTER.  IF NO, EXIT.... CSP14710
001E 0  71FF          MOX   1 -1     DECREMENT THE FIELD WIDTH           CSP14720
001F 0  70FA          MDX      LD1   NOT OONE - GET NEXT WORD            CSP14730
                    *                   EXIT.............................. CSP14740
0020 00 65000000    SAVE1 LDX  L1 *-*  DONE - RESTORE IR1                CSP14750
0022 00 4C000000    OONE1 BSC  L  *-*  RETURN TO CALLING PROGRAM          CSP14760
0024                      ENO                                           CSP14770

    NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                                  CSP14780

*STORE      WS  UA  MOVE                                               CSP14790

3382 0003
```

```
// ASM                                                                  CSP14800
** MPY SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP14810
* NAME MPY                                                        (IO) CSP14820
* LIST                                                                 CSP14830
0000     145E8000           ENT     MPY  MPY SUBROUTINE ENTRY POINT     CSP14840
                     *          CALL MPY(JCARD,J,JLAST,KCARD,K,KLAST,NER) CSP14850
                     *               THE WORDS JCARD(J) THROUGH      CSP14860
                     *               JCARD(JLAST) MULTIPLY THE WORDS CSP14870
                     *               KCARD(K) THROUGH KCARD(KLAST).  CSP14880
                     *               THE RESULT IS IN THE KCARD FIELD CSP14890
                     *               EXTENDED TO THE LEFT.           CSP14900
0000 0   0000       MPY   DC        *-*  ARGUMENT ADDRESS COMES IN HERE CSP14910
0001 0   6A6A             STX   2 SAVE2+1 SAVE IR2                     CSP14920
0002 0   696B             STX   1 SAVE1+1 SAVE IR1                     CSP14930
0003 01  65800000         LDX   I1 MPY  PUT ARGUMENT ADDRESS IN IR1     CSP14940
0005 0   C104             LD    1 4    GET K ADDRESS                   CSP14950
0006 0   D05E             STO     K1   STORE FOR FILL OF ZEROES        CSP14960
                     *               CALCULATE  K=1                   CSP14970
0007 01  C4800065         LD    I  K1  GET VALUE OF K                  CSP14980
0009 0   900B             S       ONE+1 SUBTRACT CONSTANT OF ONE       CSP14990
000A 0   D0F5             STO     MPY  STORE IN MPY                    CSP15000
000B 0   C100             LD    1 0    GET JCARD ADDRESS               CSP15010
000C 00  95800002         S     I1 2    SUBTRACT JLAST VALUE           CSP15020
000E 0   D04E             STO     SRCH+1 SAVE FOR JFRST SEARCH         CSP15030
000F 0   D075             STO     MULT1+1 SAVE FOR MULTIPLICATION      CSP15040
0010 0   8004             A       ONE+1 ADD CONSTANT OF ONE            CSP15050
0011 0   D02F             STO     OK+2 CREATE ADDRESS OF JCARD(JLAST)  CSP15060
0012 00  C5800002   TWO   LD    I1 2    GET JLAST VALUE                CSP15070
0014 00  95800001   ONE   S     I1 1    SUBTRACT J VALUE               CSP15080
0016 0   80FE             A       ONE+1 ADD CONSTANT OF ONE            CSP15090
0017 0   4808             BSC     +    CHECK FIELD WIDTH               CSP15100
0018 0   C0FC             LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE   CSP15110
0019 0   D024             STO     SCHCT+1 SAVE FIELD WIDTH FOR SEARCH  CSP15120
001A 0   C103             LD    1 3    GET KCARD ADDRESS               CSP15130
001B 0   D03C             STO     KCRD1 SAVE FOR FILL                  CSP15140
001C 0   D047             STO     KCRD2 SAVE FOR FILL                  CSP15150
001D 0   D074             STO     KCRD3 SAVE FOR CARRY                 CSP15160
001E 00  95800005         S     I1 5    SUBTRACT JLAST VALUE           CSP15170
0020 0   0054             STO     PICK+1 SAVE FOR MULTIPLICATION       CSP15180
0021 0   D059             STO     PUT1+1 SAVE FOR MULTIPLICATION       CSP15190
0022 0   80F2             A       ONE+1 ADD CONSTANT OF ONE            CSP15200
0023 0   D027             STO     SGNK+1 CREATE ADDRESS OF KCARD(KLAST) CSP15210
0024 0   C105             LD    1 5    GET KLAST ADDRESS               CSP15220
0025 0   006E             STO     KLAS2 SAVE FOR CARRY                 CSP15230
0026 0   D03F             STO     KLAS1 SAVE FOR FILL                  CSP15240
0027 00  C5800005         LD    I1 5    GET KLAST VALUE                CSP15250
0029 00  95800004         S     I1 4    SUBTRACT K VALUE               CSP15260
002B 0   80E9             A       ONE+1 ADD CONSTANT OF ONE            CSP15270
002C 0   4808             BSC     +    CHECK FIELD WIDTH               CSP15280
002D 0   C0E7             LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE   CSP15290
002E 0   D043             STO     MULTC+1 SAVE FOR MULTIPLICATION      CSP15300
002F 0   7107             MDX   1 7    MOVE OVER SEVEN ARGUMENTS       CSP15310
0030 0   693F             STX   1 DONE1+1 CREATE RETURN ADDRESS        CSP15320
                     *               KSTRT=K-JLAST+J-1                CSP15330
0031 0   C0CE             LD      MPY  LOAD K-1                        CSP15340
0032 00  8580FFFA         A     I1 -6   ADD VALUE OF J                 CSP15350
0034 00  9580FFFB         S     I1 -5   SUBTRACT VALUE OF JLAST        CSP15360
```

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

```
0036 01 4C30003D        BSC  L  SCHCT+-Z IF KSTRT PDSITIV-GO TO SCHCT CSP15370
                *                NER=KLAST                            CSP15380
0038 00 C580FFFE        LD   I1 -2   NOT POSITIVE-LOAD KLAST VALUE   CSP15390
003A 00 0580FFFF  MONE  STD  I1 -1   ANO STORE AT NER                CSP15400
003C 0  7030            MDX     SAVE1 GO TO EXIT                     CSP15410
                *                JFRST=J                              CSP15420
003D 00 65000000  SCHCT LOX  L1 *-*  LOAD IR1 WITH JCARO FIELO WIOTH  CSP15430
003F 0  D0FE      DK    STO     SCHCT+1 SAVE XSTRT IN SCHCT+1        CSP15440
                *                CLEAR AND SAVE THE SIGNS ON THE      CSP15450
                *                JCARO AND THE KCARD FIELDS           CSP15460
0040 00 C4000000        LD   L  *-*  GET JCARD(JLAST) VALUE           CSP15470
0042 0  005C            STO     JSIGN SAVE SIGN IN JSIGN             CSP15480
0043 01 4C100049        BSC  L  OVRJ+- IF NOT NEGATIVE-GO TO OVRJ    CSP15490
0045 0  F0F5            EOR     MONE+1 NEGATIVE-MAKE SIGN POSITIVE   CSP15500
0046 01 04800041        STD  I  DX+2 ANO PUT BACK IN JCARD(JLAST)    CSP15510
0048 0  C0F2            LD      MONE+1 PICXUP A MINUS ONE            CSP15520
0049 0  1890      OVRJ  SRT     16   PUT JSIGN INOICATION IN EXTENTON CSP15530
004A 00 C4000000  SGNK  LD   L  *-*  PICKUP KCARO(KLAST)             CSP15540
004C 01 4C100054        BSC  L  KPLUS+- IF NOT NEGATIVE-GO TO KPLUS  CSP15550
004E 0  F0EC            EOR     MONE+1 NEGATIVE-MAKE POSITIVE ANO    CSP15560
004F 01 D480004B        STD  I  SGNK+1 PUT 8ACK IN KCARO(KLAST)      CSP15570
0051 0  1090            SLT     16   GET JSIGN INDICATION            CSP15580
0052 0  F0E8            EOR     MONE+1 CHANGE IT                     CSP15590
0053 0  7001            MOX     DVRX SKIP THE NEXT INSTRUCTION       CSP15600
0054 0  1090      XPLUS SLT     16   GET JSIGN INDICATION            CSP15610
0055 0  D04A      OVRK  STO     XSIGN SAVE SIGN FOR RESULT           CSP15620
                *                FILL LEFT EXTENSION OF KCARO         CSP15630
                *                WITH ZEROES                          CSP15640
0056 30 062534C0        CALL    FILL FILL KCARD EXTENSION WITH ZEROES CSP15650
0058 0  0000      KCR01 DC      *-*  AODRESS OF KCARD                 CSP15660
0059 1  003E            DC      SCHCT+1 AOORESS OF KSTRT             CSP15670
005A 1  0000            DC      MPY  ADORESS DF K-1                  CSP15680
005B 1  00A1            DC      ZIP  AODRESS DF ZERO                 CSP15690
                *                IS JCARD(JLAST) PDSITIVE             CSP15700
005C 00 C5000000  SRCH  LO   L1 *-*  PICKUP JCARD(JFRST)             CSP15710
005E 01 4C3000T1        BSC  L  MULTC+-Z IF PDSITIVE-GO TO MULTC     CSP15720
                *                SEE IF JFRST IS LESS THAN JLAST.     CSP15730
                *                IF YES, JFRST=JFRST+1 AND GO         CSP15740
                *                BACK FOR MORE.  IF NO,               CSP15750
                *                MULTIPLICATION IS 8Y ZERO.           CSP15760
0060 0  T1FF            MOX   1 -1   NOT POSITIVE-DECREMENT IR1      CSP15770
0061 0  T0FA            MOX     SRCH NOT DONE - GO 8ACK FOR MORE     CSP15780
                *                FILL WITH ZERO SINCE MULTIPLIER      CSP15790
                *                IS ZERO                              CSP15800
0062 30 062534C0        CALL    FILL DONE-MAKE ENTIRE RESULT ZERO    CSP15810
0064 0  0000      KCR02 OC      *-*  AODRESS OF KCARO                 CSP15820
0065 0  0000      K1    OC      *-*  ADDRESS DF K                     CSP15830
0066 0  0000      KLAS1 DC      *-*  ADDRESS DF XLAST                 CSP15840
0067 1  00A1            OC      ZIP  ADDRESS DF ZERO                 CSP15850
                *                RESTORE THE SIGN OF JCARD            CSP15860
                *                EXIT.............................    CSP15870
0068 0  C036      FIN   LD      JSIGN PICKUP JCARD SIGN              CSP15880
0069 01 04800041        STO  I  OX+2 AND RESTORE IT                  CSP15890
006B 00 66000000  SAVE2 LDX  L2 *-*  RESTORE IR2                     CSP15900
006D 00 65000000  SAVE1 LDX  L1 *-*  RESTORE IR1                     CSP15910
```

```
006F 00 4C000000     DONE1 BSC  L  *-*  RETURN TO CALLING PROGRAM          CSP15920
                     *                   KM=K                             CSP15930
0071 00 66000000     MULTC LDX  L2 *-*  POSITIVE-LOAD IR2 WITH KCARD CNT  CSP15940
0073 0  69F1               STX   1 K1   SAVE JFRST AT K1                  CSP15950
                     *                   MULT=KCARD(KM)                   CSP15960
0074 00 C6000000     PICK  LD   L2 *-*  PICKUP KCARD(KM)                  CSP15970
0076 01 4C08008E           BSC  L  MO,+ IS IT POSITIVE-NO-GO TO MO        CSP15980
0078 0  D0ED               STO     KLAS1 YES-SAVE KCARD(KM)               CSP15990
0079 0  1810               SRA     16    CLEAR ACCUMULATOR                CSP16000
                     *                   KCARD(KM)=0                      CSP16010
007A 00 D6000000     PUT1  STO  L2 *-*  SET KCARD(KM)=0                   CSP16020
                     *                   KNOW=KM+JFRST-JLAST              CSP16030
007C 0  6AF5               STX   2 MULTC+1 GET THE VALUE                  CSP16040
007D 0  C0F4               LD      MULTC+1 OF KM                          CSP16050
007E 0  80E6               A       K1    AND ADD JFRST                    CSP16060
007F 0  80BB               A       MONE+1 TO IT AND CALCULATE             CSP16070
0080 0  80FA               A       PUT1+1 THE ADDRESS OF                  CSP16080
0081 0  D007               STO     PUT2+1 KCARD(KNOW)                     CSP16090
                     *                   JNOW=JFRST                       CSP16100
0082 01 65800065           LDX  I1 K1   LOAD IR1 WITH JFRST               CSP16110
                     *                   KCARD(KNOW)=MULT*JCARD(JNOW)     CSP16120
                     *                                +KCARD(KNOW)        CSP16130
0084 00 C5000000     MULT1 LD   L1 *-*  PICKUP JCARD(JNOW)                 CSP16140
0086 0  A0DF               M       KLAS1 MULTIPLY BY MULT                 CSP16150
0087 0  1090               SLT     16    RE-ALIGN THE PRODUCT             CSP16160
0088 00 D4000000     PUT2  STO  L  *-*  STORE IN KCARD(KNOW)               CSP16170
                     *                   KNOW=KNOW+1                      CSP16180
008A 01 74FF0089           MDX  L  PUT2+1,-1 MODIFY ADDR OF KCARD(KNOW)   CSP16190
                     *                   SEE IF JNOW IS LESS THAN JLAST.  CSP16200
                     *                   IF YES, JNOW=JNOW+1 AND GO BACK  CSP16210
                     *                   FOR MORE.  IF NO, CHECK KM.      CSP16220
008C 0  71FF               MDX   1 -1   DECREMENT IR1                     CSP16230
008D 0  70F6               MDX     MULT1 NOT DONE-GO BACK FOR MORE        CSP16240
                     *                   SEE IF KM IS LESS THAN KLAST.    CSP16250
                     *                   IF YES, KM=KM+1 AND GO BACK FOR  CSP16260
                     *                   MORE.  IF NO, RESOLVE CARRIES.   CSP16270
008E 0  72FF         MO    MDX   2 -1   DONE-DECREMENT IR2                CSP16280
008F 0  70E4               MDX     PICK NOT DONE-GO BACK FOR MORE         CSP16290
                     *                   RESOLVE CARRIES IN THE PRODUCT   CSP16300
0090 30 03059668           CALL    CARRY DONE-RESOLVE CARRIES IN THE RES  CSP16310
0092 0  0000         KCRD3 DC      *-*  ADDRESS OF KCARD                   CSP16320
0093 1  003E               DC      SCHCT+1 ADDRESS OF KSTRT               CSP16330
0094 0  0000         KLAS2 DC      *-*  ADDRESS OF KLAST                   CSP16340
0095 1  0092               DC      KCRD3 DUMMY                            CSP16350
                     *                   GENERATE THE SIGN OF THE PRODUCT CSP16360
0096 0  C009               LD      KSIGN PICKUP THE SIGN INDICATOR        CSP16370
0097 01 4C100068           BSC  L  FIN,- IF NOT NEGATIVE-ALL DONE-EXIT    CSP16380
0099 01 C480004B           LD   I  SGNK+1 NEGATIVE-PICKUP KCARD(KLAST)    CSP16390
009B 0  F09F               EOR     MONE+1 CHANGE THE SIGN                 CSP16400
009C 01 D480004B           STO  I  SGNK+1 RESTORE KCARD(KLAST)            CSP16410
009E 0  70C9               MDX     FIN   GO TO EXIT                       CSP16420
009F 0  0000         JSIGN DC      *-*  SIGN OF JCARD                     CSP16430
00A0 0  0000         KSIGN DC      *-*  SIGN OF PRODUCT                   CSP16440
00A1 0  0000         ZIP   DC      0    CONSTANT OF ZERO                  CSP16450
00A2               END                                                    CSP16460

   NO ERRORS IN ABOVE ASSEMBLY.

// DUP                                                                    CSP16470
*STORE      WS  UA  MPY                                                   CSP16480
3385 000A
```

```
// ASM                                                                  CSP16490
** NCOMP SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP16500
* NAME NCOMP                                                        (ID) CSP16510
* LIST                                                                 CSP16520
0000     150D6517           ENT     NCOMP NCOMP SUBROUTINE ENTRY POINT  CSP16530
                       *                NCOMP(JCARD,J,JLAST,KCARD,K)       CSP16540
                       *                THE WORDS JCARD(J) THROUGH         CSP16550
                       *                JCARD(JLAST) STARTING WITH         CSP16560
                       *                JCARD(J) ARE COMPARED LOGICALLY    CSP16570
                       *                TO THE FIELD STARTING AT           CSP16580
                       *                KCARD(K).  ALL DATA MUST BE IN     CSP16590
                       *                A1 FORMAT.                         CSP16600
0000 0  0000           NCOMP DC      *-*  ARGUMENT ADDRESS COMES IN HERE    CSP16610
0001 0  6925                 STX     1 SAVE1+1 SAVE IR1                    CSP16620
0002 01 65800000             LDX     I1 NCOMP PUT ARGUMENT ADDRESS IN IR1  CSP16630
0004 0  C100                 LD      1 0     GET JCARD ADDRESS             CSP16640
0005 00 95800002             S       I1 2    SUBTRACT JLAST VALUE          CSP16650
0007 0  D017                 STO       LD1+1 CREATE END OF JCARD ADDRESS   CSP16660
0008 00 C5800002             LD      I1 2    GET JLAST VALUE               CSP16670
000A 00 95800001     ONE     S       I1 1    SUBTRACT J VALUE              CSP16680
000C 0  4828                 BSC       +Z    CHECK FIELD WIDTH             CSP16690
000D 0  1810                 SRA       16    NEGATIVE = MAKE IT ZERO       CSP16700
000E 0  D00A                 STO       LDX+1 SAVE FIELD WIDTH              CSP16710
000F 0  C103                 LD      1 3     GET KCARD ADDRESS             CSP16720
0010 00 95800004             S       I1 4    SUBTRACT K VALUE              CSP16730
0012 0  9006                 S         LDX+1 SUBTRACT FIELD WIDTH          CSP16740
0013 0  D007                 STO       LD2+1 CREATE END OF KCARD ADDRESS   CSP16750
0014 01 74010019             MDX     L LDX+1,1 MAKE FIELD WIDTH TRUE       CSP16760
0016 0  7105                 MDX     1 5     MOVE OVER FIVE ARGUMENTS      CSP16770
0017 0  6911                 STX     1 DONE1+1 CREATE RETURN ADDRESS       CSP16780
                       *                JNOW=J                             CSP16790
                       *                KNOW=K+JNOW-J                      CSP16800
0018 00 65000000     LDX     LDX     L1 *-*  PUT FIELD WIDTH IN IR1         CSP16810
001A 00 C5000000     LD2     LD      L1 *-*  PICKUP JCARD(JNOW)             CSP16820
001C 0  1804                 SRA       4     DIVIDE BY EIGHT               CSP16830
001D 0  D0FB                 STO       LDX+1 SAVE TEMPORARILY              CSP16840
001E 00 C5000000     LD1     LD      L1 *-*  PICKUP KCARD(KNOW)            CSP16850
0020 0  1804                 SRA       4     DIVIDE BY EIGHT               CSP16860
0021 0  90F7                 S         LDX+1 CALCUL JCARD(JNOW)-KCARD(KNOW) CSP16870
0022 01 4C200026             BSC     L SAVE1,2 IS NCOMP ZERO-NO-ALL DONE   CSP16880
                       *                SEE IF JNOW IS LESS THAN JLAST.    CSP16890
                       *                IF YES, JNOW=JNOW+1 AND GO BACK    CSP16900
                       *                FOR MORE.  IF NO, EXIT.            CSP16910
0024 0  71FF                 MDX     1 -1    YES=DECREMENT FIELD WIDTH     CSP16920
0025 0  70F4                 MDX       LD2   GO BACK FOR MORE              CSP16930
                       *                ALL DONE - - EXIT.................. CSP16940
0026 00 65000000     SAVE1 LDX     L1'*-*  RESTORE IR1                     CSP16950
0028 00 4C000000     DONE1 BSC     L  *-*  RETURN TO CALLING PROGRAM        CSP16960
002A                       END                                            CSP16970

    NO ERRORS IN ABOVE ASSEMBLY.

// DUP                                                                  CSP16980
*STORE      WS  UA  NCOMP                                               CSP16990
338F 0004
```

```
// ASM                                                                  CSP17000
** NSIGN SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP17010
* NAME NSIGN                                                        (ID) CSP17020
* LIST                                                                 CSP17030
0000     158891D5          ENT     NSIGN NSIGN SUBROUTINE ENTRY POINT  CSP17040
                     *                    CALL NSIGN(JCARD,J,NEWS,NOLDS)  CSP17050
                     *                    THE SIGN OF THE DIGIT AT        CSP17060
                     *                    JCARD(J) IS TESTED AND NOLDS IS CSP17070
                     *                    SET.  THE SIGN IS MODIFIED AS   CSP17080
                     *                    INDICATED BY NEWS.              CSP17090
0000 0  0000      NSIGN DC      *-*   ARGUMENT ADDRESS COMES IN HERE      CSP17100
0001 0  691A            STX   1 SAVE1+1 SAVE IR1                          CSP17110
0002 01 65800000        LDX   I1 NSIGN PUT ARGUMENT ADDRESS IN IR1        CSP17120
0004 0  C100            LD    1 0     GET JCARD ADDRESS                   CSP17130
0005 00 95800001  ONE   S     I1 1    SUBTRACT J VALUE                    CSP17140
0007 0  80FE            A         ONE+1 ADD CONSTANT OF ONE               CSP17150
0008 0  D001            STO       CHAR+1 CREATE JCARD(J) ADDRESS          CSP17160
                     *                    JTEST=JCARD(J)                  CSP17170
0009 00 C4000000  CHAR  LD    L   *-*   PICKUP DIGIT                      CSP17180
000B 01 4C10001F        BSC   L   PLUS,- IS JTEST NEGATIV-NO-GO TO PLUS   CSP17190
0000 0  1890            SRT       16    YES-SAVE TEMPORARILY              CSP17200
                     *                    NOLDS=-I                        CSP17210
000E 0  C019            LD        HFFFF PICKUP MINUS ONE                  CSP17220
000F 00 D5800003        STO   I1  3     STORE IN NOLDS                    CSP17230
                     *                    NEWS*JTEST IS COMPARED TO 2ERO  CSP17240
                     *                    NEWS IS COMPARED TO ZERO        CSP17250
0011 00 C5800002        LD    I1  2     PICKUP NEWS                       CSP17260
0013 01 4C280019        BSC   L   FIN,+2 IF NEGATIVE ALL DONE             CSP17270
                     *                    JTEST=-JTEST-1                  CSP17280
0015 0  1090      REV   SLT       16    RESTORE JTEST                     CSP17290
0016 0  F011            EOR       HFFFF CHANGE THE SIGN                   CSP17300
                     *                    JCARD(J)=JTEST                  CSP17310
0017 01 D480000A        STO   I   CHAR+1 PUT NEW SIGN IN JCARD(J)         CSP17320
0019 0  7104      FIN   MDX     I 4     MOVE OVER FOUR ARGUMENTS          CSP17330
001A 0  6903            STX     I DONE1+1 CREATE RETURN ADDRESS           CSP17340
                     *                    EXIT.......................... CSP17350
001B 00 65000000  SAVE1 LDX   L1  *-*   RESTORE IR1                      CSP17360
001D 00 4C000000  DONE1 BSC   L   *-*   RETURN TO CALLING PROGRAM          CSP17370
001F 0  1890      PLUS  SRT       16    SAVE TEMPORARILY                  CSP17380
                     *                    NOLDS=I                         CSP17390
0020 0  C0E5            LD        ONE+1 PICKUP CONSTANT OF ONE            CSP17400
0021 00 D5800003        STO   I1  3     STORE IT IN NOLDS                 CSP17410
                     *                    NEWS*JTEST IS COMPARED TO 2ERO  CSP17420
                     *                    NEWS IS COMPARED TO ZERO        CSP17430
0023 00 C5800002        LD    I1  2     PICKUP NEWS                       CSP17440
0025 01 4C300019        BSC   L   FIN,-2 IF POSITIVE - ALL DONE           CSP17450
0027 0  70ED            MDX       REV   REVERSE SIGN - GO TO REV          CSP17460
0028 0  FFFF      HFFFF DC        /FFFF CONSTANT OF MINUS ONE             CSP17470
002A                    END                                               CSP17480

   NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                                  CSP17490
*STORE      WS  UA  NSIGN                                              CSP17500
3393 0004
```

```
// ASM                                                                  CSP1751D
** NZONE SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP175Z0
* NAME NZONE                                                         (ID) CSP17530
* LIST                                                                 CSP17540
0000    15A56545          ENT     NZONE NZONE SUBROUTINE ENTRY POINT   CSP17550
                     *                 CALL NZONE(JCARD,J,NEWZ,NOLDZ)   CSP17560
                     *                 THE ZONE OF THE CHARACTER AT     CSP17570
                     *                 JCARD(J) IS TESTED AND NOLDZ IS  CSP17580
                     *                 SET.  THE ZONE IS MODIFIED AS    CSP17590
                     *                 INDICATED BY NEWZ.               CSP17600
0000 0  0000       NZONE DC      *-*  ARGUMENT ADDRESS COMES IN HERE     CSP17610
0001 0  69Z5              STX   1 SAVE1+1 SAVE IR1                       CSP176Z0
0002 01 65800000          LDX   I1 NZONE PUT ARGUMENT ADDRESS IN IR1     CSP17630
0004 0  C100              LD    1 0     GET JCARD ADDRESS                CSP17640
0005 00 95800001   ONE    S     I1 1    SUBTRACT J VALUE                 CSP17650
0007 0  80FE              A         ONE+1 ADD CONSTANT OF ONE            CSP17660
0008 0  D01A              STO       STO+1 CREATE JCARD(J) ADDRESS        CSP17670
0009 0  D001              STO       LD1+1 CREATE JCARD(J) ADDRESS        CSP17680
                     *                 JTEST=JCARD(J)                   CSP17690
000A 00 C4000000   LD1    LD    L *-*  PICKUP THE CHARACTER              CSP17700
000C 0  D0FE              STO       LD1+1 SAVE IT TEMPORARILY            CSP17710
                     *                   IS JTEST NEGATIVE              CSP177Z0
000D 01 4C10003A          BSC   L PLUS,-  IF NOT NEGATIVE-GO TO PLUS     CSP17730
000F 0  9018              S         ZERO NEGATIVE-CHECK TO SEE IF IT IS  CSP17740
0010 01 4C1800ZE          BSC   L TWO,+- AN E8CDIC ZERO-YES-GO TO TWO    CSP17750
                     *                 NOLDZ=5+(JTEST-4096)/4096        CSP17760
                     *                 SHIFT I2 IS EQUIVALENT TO DIVIDE CSP17770
                     *                 BY 4096                          CSP17780
                     *                 AND 3000 IS EQUIVALENT TO        CSP17790
                     *                 SUBTRACT 4096 AND SHIFT          CSP17800
001Z 0  C0F8              LD        LD1+1 NO-RELOAD JTEST                CSP17810
0013 0  E019              AND       H3000 REMOVE ALL BUT BITS Z AND 3    CSP178Z0
0014 0  180C              SRA       1Z   PUT IN LOW ORDER OF ACCUMULATOR CSP17830
0015 0  80F0              A         ONE+1 ADD CONSTANT OF ONE            CSP17840
0016 00 D5800003          STO   I1 3      STORE IN NOLDZ                 CSP17850
                     *                   IS NEWZ LESS THAN FIVE         CSP17860
0018 00 C580000Z          LD    I1 Z      PICKUP VALUE OF NEWZ           CSP17870
001A 0  9011              S         FOUR AND CHECK FOR LESS THAN FIVE    CSP17880
0018 01 4C300024          BSC   L FINIS,-Z NO-GO TO EXIT                 CSP17890
0010 0  800E              A         FOUR YES - RESTORE NEWZ              CSP17900
                     *                 JCARD(J)=JTEST+4096*(NEWZ-NOLDZ) CSP17910
001E 00 95800003          S     I1 3      SUBTRACT NOLDZ                 CSP179Z0
00Z0 0  100C              SLA       1Z   PUT RESULT IN BITS 2 AND 3       CSP17930
00Z1 0  80E9              A         LD1+1 ADD ORIGINAL CHARACTER         CSP17940
00ZZ 00 D4000000   STO    STO   L *-*  STORE BACK IN JCARD(J)             CSP17950
                     *                 EXIT,.....                       CSP17960
00Z4 0  7104       FINIS  MDX    1 4    MOVE OVER FOUR ARGUMENTS         CSP17970
00Z5 0  6903              STX   1 QONE1+1 CREATE RETURN ADDRESS          CSP17980
00Z6 00 65000000   SAVE1  LDX   L1 *-*  RESTORE IR1                      CSP17990
00Z8 00 4C000000   DONE1  BSC   L  *-*  RETURN TO CALLING PROGRAM         CSP18000
00ZA 0  6040       MINUS  DC        /6040 CONSTANT OF E8COIC MINUS SIGN  CSP18010
00ZB 0  F040       ZERO   DC        /F040 CONSTANT OF E8CDIC ZERO        CSP180Z0
00ZC 0  0004       FOUR   DC        4     CONSTANT OF FOUR               CSP18030
00ZD 0  3000       H3000  DC        /3000 CONSTANT FOR STRIPING BITS     CSP18040
                     *                 IS NEWZ TWO                      CSP18050
00ZE 00 C5800002   TWO    LD    I1 Z      PICKUP VALUE OF NEWZ           CSP18060
0030 0  90FE              S         TWO+1 IS IT TWO                      CSP18070
```



```
0031 01 4CZ00036          BSC   L NOT,Z NO - GO TO NOT                  CSP18080
                     *                 JCARD(J)=Z4640                   CSP18090
0033 0  C0F6              LD        MINUS YES - SET JCARD(J)             CSP18100
0034 01 D4800023          STO   I STO+1 EQUAL TO AN E8CDIC MINUS SIGN    CSP18110
                     *                 NOLDZ=4                          CSP181Z0
0036 0  C0F5       NOT    LD        FOUR SET NOLDZ                       CSP18130
0037 00 D5800003          STO   I1 3      EQUAL TO FOUR                  CSP18140
0039 0  70EA              MDX       FINIS GO TO EXIT                     CSP18150
                     *                   IS JTEST AN EBCDIC MINUS SIGN  CSP18160
003A 0  90EF       PLUS   S         MINUS NOT NEGATIVE - CHECK FOR E8CDIC CSP18170
0038 01 4CZ00049          BSC   L SPEC,Z MINUS SIGN-NO-GO TO SPEC        CSP18180
                     *                 NOLDZ=Z                          CSP18190
003D 0  C0F1              LD        TWO+1 YES-LOAD TWO AND STORE         CSP182D0
003E 00 D5800003          STO   I1 3      IT IN NOLDZ                    CSP18Z10
                     *                   IS NEWZ FOUR                   CSP18ZZ0
0040 00 C5800002          LD    I1 Z      PICKUP VALUE OF NEWZ AND       CSP18Z30
004Z 0  90E9              S         FOUR CHECK FOR VALUE OF FOUR         CSP18Z40
0043 01 4C200024          BSC   L FINIS,Z NO-GO TO FINIS                 CSP18Z50
                     *                 JCARD(J)=-403Z                   CSP18Z60
0045 0  C0E5              LD        ZERO YES-LOAD EBCDIC ZERO AND        CSP18Z70
0046 01 D48000Z3          STO   I STO+1 STORE IT AT JCARD(J)             CSP18Z80
0048 0  70D8       BIG    MDX       FINIS GO TO EXIT                     CSP18Z90
0049 0  C0FE       SPEC   LD        BIG  SPECIAL CHARACTER-LOAD LARGE    CSP18300
004A 00 D5800003          STO   I1 3      NUMBER AND STORE AT NOLDZ      CSP18310
004C 0  70D7              MDX       FINIS ALL DONE - GO TO EXIT          CSP183Z0
004E                      END                                            CSP18330
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP18340
*STORE      WS  UA  NZONE                                              CSP18350
3397 0006
```

```
// ASM                                                                  CSP18360
** PRINT AND SKIP SUBRDUTINES FOR 1130 CSP                        (ID) CSP18370
* NAME PRINT                                                      (ID) CSP18380
* LIST                                                                CSP18390
0041    17649563          ENT     PRINT     SUBRDUTINE ENTRY PDINT    CSP18400
                  * CALL PRINT (JCARD, J, JLAST, NERR3)               CSP18410
                  * PRINT JCARD(J) THROUGH JCARD(JLAST) ON THE        CSP18420
                  * 1132 PRINTER. PUT ERROR PARAMETER IN NERR3.       CSP18430
0069    224B95C0          ENT     SKIP      SUBRDUTINE ENTRY PDINT    CSP18440
                  * CALL SKIP(N)                                      CSP18450
                  * EXECUTE CDNTRDL FUNCTIDN SPECIFIED BY INTEGER N   CSP18460
0000 0  0001      ONE     DC      1         CDNSTANT DF 1             CSP18470
0001 0  2000      SPACE   DC      /2000     PRINT FUNCTIDN WITH SPACE CSP18480
0002 0  0000      JCARD   DC      *-*       JCARD J  ADDRESS          CSP18490
0003 0  0000      JLAST   DC      *-*       JCARD JLAST  ADDRESS      CSP18500
0004    003D      AREA    BSS     61        WDRD CDUNT & PRINT AREA   CSP18510
0041 0  0000      PRINT   DC      *-*       ADDRESS DF 1ST ARGUMENT   CSP18520
0042 20 17655BF1  TEST    LIBF    PRNT1     CALL BUSY TEST RDUTINE    CSP18530
0043 0  0000              DC      /0000     BUSY TEST PARAMETER       CSP18540
0044 0  70FD              MDX     TEST      REPEAT TEST IF BUSY       CSP18550
0045 0  691A              STX   1 SAVE1&1   STDRE IR1                 CSP18560
0046 01 65800041          LDX   I1 PRINT    LDAD 1ST ARGUMENT ADDRESS CSP18570
004B 20 01647BB0          LIBF    ARGS      CALL ARGS RDUTINE         CSP18580
0049 1  0002              DC      JCARD     JCARD J  PICKED UP        CSP18590
004A 1  0003              DC      JLAST     JCARD JLAST  PICKED UP   CSP18600
004B 1  0004              DC      AREA      CHARACTER CDUNT PICKED UP CSP18610
004C 0  0078              DC      120       MAX CHARACTER COUNT       CSP18620
0040 0  C0B6              LD      AREA      GET CHARACTER COUNT       CSP18630
004E 0  B0B1              A       DNE       HALF ADJUST               CSP18640
004F 0  1B01              SRA     1         DIVIDE BY TWO             CSP18650
0050 0  D0B3              STD     AREA      STDRE WDRD COUNT          CSP18660
0051 0  C103              LD    1 3         GET ERRDR WDRD ADDRESS    CSP18670
0052 0  D012              STD     ERR&1     STORE IT IN ERRDR RDUTINE CSP18680
0053 20 195C10D2          LIBF    RPACK     CALL REVERSE PACK RDUTINE CSP18690
0054 1  0002              DC      JCARD     JCARD J  ADDRESS          CSP18700
0055 1  0003              DC      JLAST     JCARD JLAST  ADDRESS      CSP18710
0056 1  0005              DC      AREA&1    PACK INTD I/D AREA        CSP18720
0057 20 17655BF1          LIBF    PRNT1     CALL PRINT RDUTINE        CSP18730
005B 0  2000      WRITE   DC      /2000     PRINT PARAMETER           CSP18740
0059 1  0004              DC      AREA      I/O AREA BUFFER           CSP18750
005A 1  0063              DC      ERRDR     ERROR PARAMETER           CSP18760
005B 0  C0A5              LD      SPACE     LDAD PRINT WITH SPACE     CSP18770
005C 0  D0FB              STD     WRITE     STORE IN PRINT PARAMETER  CSP18780
0090 0  7104              MDX   1 4         INCREMENT DVER 4 ARGUMENTS CSP18790
005E 0  6903              STX   1 DONE1&1   STDRE IR1                 CSP18800
005F 00 65000000  SAVE1   LDX   L1 *-*      RELDAD DR RESTORE IR1     CSP18810
0061 00 4C000000  DONE1   BSC   L  *-*      RETURN TD CALLING PRDGRAM CSP18820
0063 0  0000      ERROR   DC      *-*       RETURN ADDRESS GOES HERE  CSP18830
0064 00 D4000000  ERR     STD   L  *-*      STDRE ACC IN ERRDR PARAM  CSP18840
0066 0  1B10              SRA     16        CLEAR ACC                 CSP18850
0067 01 4C800063          BSC   I  ERRDR    RETURN TD PRNT1 PRDGRAM   CSP18860
0069 0  0000      SKIP    DC      *-*       ADDRESS DF ARGUMENT ADDR  CSP18870
006A 01 C4800069          LD    I  SKIP     GET ARGUMENT ADDRESS      CSP18880
006C 0  0001              STO     ARG&1     DROP IT AND               CSP18890
0060 00 C4000000  ARG     LD    L  *-*      GET ARGUMENT              CSP18900
006F 01 4C300074          BSC   L  NDSUP,-2 GD TD NDSUPPRESSION IF &  CSP18910
0071 0  C009              LD      NOSPC     SET UP SPACE SUPPRESSION  CSP18920
```



```
0072 0  D0E5              STO     WRITE     CHANGE PRINT FUNCTION     CSP18930
0073 0  7003              MDX     DDNE      GD TD RETURN              CSP18940
0074 0  D001      NDSUP   STD     CNTRL     SET UP COMMAND            CSP18950
0075 20 17655BF1          LIBF    PRNT1     CALL THE PRNT RDUTINE     CSP18960
0076 0  3000      CNTRL   DC      /3000     CARRIAGE CDMMAND WDRD     CSP18970
0077 01 74010069  DDNE    MDX   L SKIP,1    ADJUST RETURN ADDRESS     CSP18980
0079 01 4CB00069          BSC   I  SKIP     RETURN TO CALLING PROGRAM CSP18990
007B 0  2010      NDSPC   DC      /2010     SUPPRESS SPACE COMMAND    CSP19000
007C                      END               END OF PRINT SUBPRDGRAM   CSP19010
```

NO ERRDRS IN ABDVE ASSEMBLY.

```
// DUP                                                                 CSP19020
*STDRE      WS  UA  PRINT                                             CSP19030
339D 0005
```

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

```
// ASM                                                                  CSP19040
** PUT SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE                (ID) CSP19050
* NAME PUT                                                              (ID) CSP19060
* LIST                                                                      CSP1907D
0000    17923000          ENT     PUT  PUT SUBROUTINE ENTRY POINT         CSP19080
                     *                   CALL PUT(JCARD,J,JLAST,VAR,ADJST,N) CSP19090
                     *                        THE REAL NUMBER VAR IS HALF-  CSP19100
                     *                    ADJUSTED WITH ADJST AND           CSP19110
                     *                    TRUNCATED.  THEN DIGITS ARE       CSP19120
                     *                    CONVERTED FROM REAL TO EBCDIC     CSP19130
                     *                    AND PLACED IN THE JCARD FIELD     CSP19140
                     *                    FROM JCARD(JLAST) TO JCARD(J).    CSP19150
0000 0  0000        PUT   DC      *-*  ARGUMENT ADDRESS COMES IN HERE       CSP19160
0001 0  6957              STX   1 FIN+1 SAVE IR1                            CSP19170
0002 01 65800000          LDX   I1 PUT   PUT ARGUMENT ADDRESS IN IR1        CSP19180
0004 0  C100              LD    1 0    GET JCARD ADDRESS                  CSP19190
0005 0  D04E              STO     JCRD1 SAVE FOR NZONE SUBROUTINE           CSP19200
0006 00 95B00002          S     I1 2    SUBTRACT JLAST VALUE               CSP19210
0008 0  B00E              A       ONE+1 ADD CONSTANT OF ONE                 CSP19220
0009 0  D03D              STO     PUT1+1 CREATE JCARD(JLAST) ADDRESS        CSP19230
000A 0  C103              LD    1 3    GET VAR ADDRESS                    CSP19240
000B 0  D014              STO     VAR  SAVE FOR PICKUP                      CSP19250
000C 0  B00A              A       ONE+1 ADD CONSTANT OF ONE                 CSP19260
000D 0  0041              STO     SIGN+1 SAVE SIGN POSITION ADDRESS         CSP19270
000E 0  C104              LD    1 4      GET ADJST ADDRESS                  CSP19280
000F 0  D012              STO     ADJST AND SAVE                            CSP19290
0010 00 C5B00005          LD    I1 5    GET N VALUE AND                    CSP19300
0012 0  D017              STO     ADRN2+1 SAVE FOR TRUNCATION               CSP19310
0013 00 C5B00002    TWO   LD    I1 2      GET JLAST VALUE AND              CSP19320
0015 0  D024              STO     JLAST SAVE IT AT JLAST                    CSP19330
0016 00 95B00001    ONE   S     I1 1    SUBTRACT J VALUE                   CSP19340
0018 0  B0FE              A       ONE+1 ADD CONSTANT OF ONE                 CSP19350
0019 0  480B              BSC     +     CHECK FIELD WIDTH                   CSP19360
001A 0  C0FC              LD      ONE+1 NEGATIVE OR ZERO-MAKE IT ONE        CSP1937D
001B 0  D017              STO     PUTCT+1 OK-SAVE FIELD WIDTH               CSP19380
001C 0  7106              MDX   1 6    MOVE OVER SIX ARGUMENTS              CSP19390
001D 0  693D              STX   1 DONE1+1 CREATE RETURN ADDRESS             CSP19400
                     *                   DIGS=WHOLE(ABS(VAR)+ADJST)        CSP1941D
001E 30 05042B80          CALL    EABS TAKE THE ABSOLUTE VALUE              CSP19420
0020 0  0000        VAR   DC      *-*  OF VAR                               CSP19430
0021 20 05044100          LIBF    EADD ADD TO IT THE                        CSP19440
0022 0  0000        ADJST DC      *-*  HALF-ADJUSTMENT VALUE                CSP1945D
0023 30 262164C5          CALL    WHOLE TRUNCATE ANY FRACTION               CSP19460
0025 0  F040        ZERO  DC      /F040 CONSTANT OF EBCDIC ZERO             CSP19470
                     *                   IS N GREATER THAN ZERO            CSP19480
0026 0  C003              LD      ADRN2+1 CHECK TO SEE IF N IS GREATER      CSP19490
0027 01 4C080032          BSC   L PUTCT,+ THAN ZERO-NO-GO TO PUTCT          CSP195D0
                     *                   JNOW=1                            CSP19510
0029 00 65000000    ADRN2 LDX   L1 *-*  YES-PUT VALUE OF N IN IR1           CSP19520
002B 20 05517A00    AGAIN LIBF    EMPY MULTIPLY BY                          CSP19530
002C 1  005C              DC      PNT1 ONE TENTH                            CSP19540
002D 30 262164C5          CALL    WHOLE TRUNCATE THE FRACTION               CSP19550
002F 0  0000              DC      0    DUMMY                                CSP19560
                     *                   SEE IF JNOW IS LESS THAN N.       CSP19570
                     *                   IF YES, JNOW=JNOW+1 AND GO BACK   CSP19580
                     *                   FOR MORE.  IF NO, START           CSP19590
                     *                   CONVERTING.                       CSP19600
```

```
0030 0  71FF              MDX   1 -1    DECREMENT N BY ONE                CSP19610
0031 0  7DF9              MDX     AGAIN NOT DONE-GO BACK FOR MORE         CSP19620
                     *                  JNOW=JLAST                        CSP19630
0032 00 65000000     PUTCT LDX   L1 *-*  DONE-PUT FIELD WIDTH IN IR1       CSP19640
0034 20 058A3580     BACK  LIBF     ESTO STORE FAC                        CSP19650
0035 1  0062               DC       DIGS IN DIGS                          CSP19660
                     *                  DIGT=WHOLE(DIGS/10.0)             CSP19670
0036 20 05517A00           LIBF     EMPY MULTIPLY BY                      CSP19680
0037 1  D05C               DC       PNT1 ONE TENTH AND                    CSP19690
0038 30 262164C5           CALL     WHOLE TRUNCATE ANY FRACTION           CSP19700
003A 0  000D       JLAST   DC       *-*  JLAST VALUE                      CSP19710
003B 20 058A3580           LIBF     ESTO STORE RESULT IN                  CSP19720
003C 1  0065               DC       DIGS1 DIGS1-SAME AS DIGT              CSP19730
                     *                  JCARD(JNOW)=256*IFIX(DIGS         CSP19740
                     *                                - 10.0*DIGT)-4032   CSP19750
                     *                  MULTIPLY BY 256 IS SAME AS SHIFT  CSP19760
                     *                  EIGHT                             CSP19770
                     *                  SUBTRACT 4032 IS SAME AS OR F040  CSP19780
003D 20 05517A00           LIBF     EMPY MULTIPLY DIGT BY                 CSP19790
003E 1  005F               DC       ETEN TEN AND                          CSP19800
003F 20 15599500           LIBF     NORM NORMALIZE THE RESULT             CSP19810
0040 20 22559000           LIBF     SNR  REVERSE THE SIGN                 CSP19820
0041 20 05044100           LIBF     EADD AND ADD IN THE                   CSP19830
0042 1  0062               DC       DIGS VALUE OF DIGS                    CSP19840
0043 20 091899C0           LIBF     IFIX FIX THE RESULT                   CSP19850
0044 0  1008               SLA      8     AND PLACE IN BITS 4-7           CSP19860
0045 0  E8DF               OR       ZERO  MAKE AN A1 CHARACTER            CSP19870
0046 00 D4000000     PUT1  STO   L  *-*  AND STORE IN JCARD(JNOW)          CSP19880
0048 20 054C4000           LIBF     ELD  SET FAC EQUAL                    CSP19890
0049 1  0065               DC       DIGS1 TO DIGS1                        CSP19900
                     *                  SEE IF JNOW IS GREATER THAN J.    CSP19910
                     *                  IF YES, JNOW=JNOW-1 AND GO BACK   CSP19920
                     *                  FOR MORE.  IF NO, SET ZONE.       CSP19930
004A 01 74010047           MDX   L  PUT1+1,1 CHANGE JCARD ADDRESS         CSP19940
004C 0  71FF               MDX    1 -1   DECREMENT COUNT                  CSP19950
004D 0  70E6               MDX      BACK NOT DONE-GO BACK FOR MORE        CSP19960
                     *                  IS VAR LESS THAN ZERO             CSP19970
004E 00 C4000000     SIGN  LD    L  *-*  DONE-PICKUP ORIGINAL SIGN         CSP19980
0050 01 4C100058           BSC   L  FIN,- IF NOT NEG-ALL DONE-GO TO EXIT  CSP19990
0052 30 15A56545           CALL     NZONE CALL N2ONE FOR ZONE SETTING     CSP20000
0054 0  0000       JCRD1 DC       *-*  ADDRESS OF JCARD                   CSP20010
0055 1  003A               DC       JLAST ADDRESS OF JLAST                CSP20020
0056 1  0014               DC       TWO+1 ADDRESS OF NEW ZONE INDICATOR   CSP20030
0057 1  0054               DC       JCRD1 DUMMY                           CSP20040
                     *                  EXIT.....................         CSP20050
0058 00 65000000     FIN   LDX   L1 *-*  RESTORE IR1                       CSP20060
005A 00 4C000000     DONE1 BSC   L  *-*  RETURN TO CALLING PROGRAM          CSP20070
005C 7D 66666666     PNT1  XFLC     0.1  CONSTANT OF ONE TENTH              CSP20080
005F 84 50000000     ETEN  XFLC     10.0 CONSTANT OF TEN POINT ZERO        CSP20090
0062     0003        DIGS  BSS      3    TEMPORARY AREA FOR GETTING A DGT  CSP20100
0065     0003        DIGS1 BSS      3    TEMPORARY AREA FOR GETTING A DGT  CSP20110
0068                       END                                            CSP20120

    NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                                    CSP20130
*STORE      WS  UA  PUT                                                   CSP20140
33A2 0007
```

```
                                                                                                                CSP20150
// ASM
** PRINT AND SKIP SUBRDUTINES FOR 1130 CSP, 1403                                                        (ID) CSP20160
* NAME P1403                                                                                            (ID) CSP20170
* LIST                                                                                                       CSP20180
0041    17C74C33            ENT     P1403       SUBROUTINE ENTRY POINT                                     CSP20190
                          * CALL P1403 (JCARD, J, JLAST, NERR3)                                            CSP20200
                          * PRINT JCARD(J) THROUGH JCARD(JLAST) ON THE                                     CSP20210
                          * 1403 PRINTER. PUT ERROR PARAMETER IN NERR3.                                    CSP20220
0072    22C74C33            ENT     S1403       SUBROUTINE ENTRY POINT                                     CSP20230
                                                                                                           CSP20240
                          * CALL S1403(N)                                                                  CSP20250
                          * EXECUTE CONTROL FUNCTION SPECIFIED BY INTEGER N                                CSP20260
0000 0  0001              ONE   DC      1           CONSTANT OF 1                                          CSP20270
0001 0  2000              SPACE OC      /2000       PRINT FUNCTION WITH SPACE                              CSP20280
0002 0  0000              JCARD DC      *-*         JCARD J  ADDRESS                                       CSP20290
0003 0  0000              JLAST DC      *-*         JCARD JLAST  ADDRESS                                   CSP20300
0004    003D              AREA  BSS     61          WORD COUNT & PRINT AREA                                CSP20310
0041 0  0000              P1403 DC      *-*         ADDRESS OF 1ST ARGUMENT                                CSP20320
0042 0  6926                    STX   1 SAVE161     STORE IR1                                              CSP20330
0043 01 65800041                LDX   I1 P1403      LOAD 1ST ARGUMENT ADDRESS                              CSP20340
0045 20 016478B0                LIBF    ARGS        CALL ARGS ROUTINE                                      CSP20350
0046 1  0002                    DC      JCARD       JCARD J  PICKED UP                                     CSP20360
0047 1  0003                    DC      JLAST       JCARD JLAST  PICKED UP                                 CSP20370
0048 1  0004                    DC      AREA        CHARACTER COUNT PICKED UP                              CSP20380
0049 0  0078                    DC      120         MAX CHARACTER COUNT                                    CSP20390
004A 0  C0B9                    LD      AREA        GET CHARACTER COUNT                                    CSP20400
004B 0  80B4                    A       ONE         HALF ADJUST                                            CSP20410
004C 0  1801                    SRA     1           DIVIDE BY TWO                                          CSP20420
004D 0  D0B6                    STO     AREA        STORE WORD COUNT                                       CSP20430
004E 0  1001                    SLA     1           DOUBLE IT = CHARACTER                                  CSP20440
004F 0  000A                    STO     CNT         COUNT AND STORE COUNT                                  CSP20450
0050 0  C103                    LD    1 3           GET ERROR WORD ADDRESS                                 CSP20460
0051 0  001C                    STO     ERR61       STORE IT IN ERROR ROUTINE                              CSP20470
0052 20 195C10D2                LIBF    RPACK       CALL REVERSE PACK ROUTINE                              CSP20480
0053 1  0002                    DC      JCARD       JCARD J  ADDRESS                                       CSP20490
0054 1  0003                    DC      JLAST       JCARD JLAST  ADDRESS                                   CSP20500
0055 1  0005                    OC      AREA61      PACK INTO I/O AREA                                     CSP20510
0056 20 29257006                LIBF    ZIPCO       CALL CONVERSION ROUTINE                                CSP20520
0057 0  0000                    DC      /0000       FROM EBCDIC TO 1403 CODES                              CSP20530
0058 1  0005                    DC      AREA+1      FROM I/O AREA                                          CSP20540
0059 1  0005                    DC      AREA+1      TO I/O AREA                                            CSP20550
005A 0  0000              CNT   DC      *-*         CHARACTER COUNT                                        CSP20560
005B 30 050978F3                CALL    EBPT3       CONVERSION TABLE FOR ZIPCO                             CSP20570
0050 20 176558F3          TEST  LIBF    PRNT3       CALL BUSY TEST ROUTINE                                 CSP20580
005E 0  0000                    DC      /0000       BUSY TEST PARAMETER                                    CSP20590
005F 0  70FD                    MDX     TEST        REPEAT TEST IF BUSY                                    CSP20600
0060 20 176558F3                LIBF    PRNT3       CALL PRINT ROUTINE                                     CSP20610
0061 0  2000              WRITE DC      /2000       PRINT PARAMETER                                        CSP20620
0062 1  0004                    DC      AREA        I/O AREA BUFFER                                        CSP20630
0063 1  006C                    DC      ERROR       ERROR PARAMETER                                        CSP20640
0064 0  C09C                    LD      SPACE       LOAD PRINT WITH SPACE                                  CSP20650
0065 0  D0FB                    STO     WRITE       STORE IN PRINT PARAMETER                               CSP20660
0066 0  7104                    MDX   1 4           INCREMENT OVER 4 ARGUMENTS                             CSP20670
0067 0  6903                    STX   1 DONE161     STORE IR1                                              CSP20680
0068 00 65000000          SAVE1 LDX   L1 *-*        RELOAD OR RESTORE IR1                                  CSP20690
006A 00 4C000000          DONE1 BSC   L  *-*        RETURN TO CALLING PROGRAM                              CSP20700
006C 0  0000              ERROR DC      *-*         RETURN ADDRESS GOES HERE                               CSP20710
0060 00 04000000          ERR   STO   L *-*         STORE ACC IN ERROR PARAM                               CSP20720
                                                                                                  PAGE   2
006F 0  1810                    SRA     16          CLEAR ACC                                              CSP20730
0070 01 4C80006C                BSC   I ERROR       RETURN TO PRNT3 PROGRAM                                CSP20740
0072 0  0000              S1403 OC      *-*         ADDRESS OF ARGUMENT ADDR                               CSP20750
0073 01 C4800072                LO    I S1403       GET ARGUMENT ADDRESS                                   CSP20760
0075 0  0001                    STO     ARG61       DROP IT AND                                            CSP20770
0076 00 C4000000          ARG   LD    L *-*         GET ARGUMENT                                           CSP20780
0078 01 4C30007D                BSC   L NOSUP,-Z    GO TO NOSUPPRESSION IF &                               CSP20790
007A 0  C009                    LO      NOSPC       SET UP SPACE SUPPRESSION                               CSP20800
007B 0  D0E5                    STO     WRITE       CHANGE PRINT FUNCTION                                  CSP20810
007C 0  7003                    MDX     DONE        GO TO RETURN                                           CSP20820
007D 0  0001              NOSUP STO     CNTRL       SET UP COMMAND                                         CSP20830
007E 20 17655BF3                LIBF    PRNT3       CALL THE PRNT3 ROUTINE                                 CSP20840
007F 0  3000              CNTRL DC      /3000       CARRIAGE COMMAND WORD                                  CSP20850
0080 01 74010072          OONE  MDX   L S1403,1     ADJUST RETURN ADDRESS                                  CSP20860
0082 01 4C800072                BSC   I S1403       RETURN TO CALLING PROGRAM                              CSP20870
0084 0  2010              NOSPC DC      /2010       SUPPRESS SPACE COMMAND                                 CSP20880
0086                            END                 END OF P1403 SUBPROGRAM                                CSP20890

    NO ERRORS IN ABOVE ASSEMBLY.

// DUP                                                                                                     CSP20900
*STORE      WS  UA  P1403
33A9 0006
```

```
// ASM                                                                  CSP20910
** PUNCH SUBROUTINE FOR 1130 CSP, 1442-5                        (ID) CSP20920
* NAME P1442                                                     (ID) CSP20930
* LIST                                                                CSP20940
0053    17C74D32          ENT     P1442      SUBROUTINE ENTRY POINT    CSP20950
                  * CALL P1442 (JCARD, J, JLAST, NERR2)                CSP20960
                  * PUNCH JCARD(J) THROUGH JCARD(JLAST) INTO THE       CSP20970
                  * BEGINNING OF A CARD. PUT ERROR PARAMETER INTO      CSP20980
                  * NERR2.                                            CSP20990
0000 0  0000      JCARD DC      *-*        JCARD J  ADDRESS          CSP21000
0001    0051      AREA  BSS     81         I/O AREA BUFFER           CSP21010
0052 0  0000      FLAG  DC      *-*        ERROR INDICATOR           CSP21020
0053 0  0000      P1442 DC      *-*        FIRST ARGUMENT ADDRESS    CSP21030
0054 0  6922            STX   1 SAVE1&1    SAVE IR1                  CSP21040
0055 01 65800053        LDX   I1 P1442     LOAD 1ST ARGUMENT ADDRESS CSP21050
0057 20 01647880        LIBF    ARGS       CALL ARGS SUBPROGRAM      CSP21060
0058 1  0000            DC      JCARD      GET JCARD(J) ADDRESS      CSP21070
0059 1  0067            DC      JLAS2      GET JCARD(JLAST) ADDRESS  CSP21080
005A 1  0001            DC      AREA       GET CHARACTER COUNT       CSP21090
005B 0  0050            DC      80         MAX CHARACTER COUNT       CSP21100
005C 0  C0A4            LD      AREA       DISTRIBUTE COUNT          CSP21110
005D 0  D00B            STO     CNT2       INTO CNT2                 CSP21120
005E 0  C103            LD    1 3          GET ERROR WORD ADDRESS    CSP21130
005F 0  D01C            STO     ERR+1      STORE INSIDE ERROR ROUTINE CSP21140
0060 0  1810            SRA     16         CLEAR ACC                 CSP21150
0061 0  D0F0            STO     FLAG       CLEAR ERROR INDICATOR     CSP21160
0062 20 22989547        LIBF    SWING      CALL REVERSE ARRAY        CSP21170
0063 1  0000            DC      JCARD      FROM JCARD J              CSP21180
0064 1  0067            DC      JLAS2      TO JCARD JLAST            CSP21190
0065 20 225C5144        LIBF    SPEED      CALL CONVERSION ROUTINE   CSP21200
0066 0  0011            DC      /0011      FROM EBCDIC TO CARD CODE  CSP21210
0067 0  0000      JLAS2 DC      *-*        FROM JCARD JLAST          CSP21220
0068 1  0002            DC      AREA&1     TO THE I/O AREA BUFFER    CSP21230
0069 0  0000      CNT2  DC      *-*        CHARACTER COUNT           CSP21240
006A 20 17543231        LIBF    PNCH1      CALL PUNCH ROUTINE        CSP21250
006B 0  2000            DC      /2000      PUNCH                     CSP21260
006C 1  0001            DC      AREA       I/O AREA BUFFER           CSP21270
006D 1  007A            DC      ERROR      ERROR PARAMETER           CSP21280
006E 20 22989547        LIBF    SWING      REVERSE THE ARRAY         CSP21290
006F 1  0000            DC      JCARD      FROM JCARD(J)             CSP21300
0070 1  0067            DC      JLAS2      TOJCARD(JLAST)            CSP21310
0071 20 17543231  TEST  LIBF    PNCH1      CALL BUSY TEST ROUTINE    CSP21320
0072 0  0000            DC      /0000      BUSY TEST PARAMETER       CSP21330
0073 0  70FD            MDX     TEST       REPEAT IF BUSY            CSP21340
0074 0  7104            MDX   1 4          INCREMENT 4 ARGUMENTS     CSP21350
0075 0  6903            STX   1 DONE+1     STORE IR1                 CSP21360
0076 00 65000000  SAVE1 LDX   L1 *-*       RESTORE IR1               CSP21370
0078 00 4C000000  DONE  BSC   L  *-*       RETURN TO CALLING PROGRAM CSP21380
007A 0  0000      ERROR DC      *-*        START OF ERROR ROUTINE    CSP21390
007B 00 D4000000  ERR   STO   L  *-*       STORE ACC IN ERROR WORD   CSP21400
007D 01 74010052        MDX   L  FLAG,1    SET THE FLAG INDICATOR    CSP21410
007F 01 4C80007A        BSC   I  ERROR     RETURN TO INTERRUPT PROGRM CSP21420
0082                    END                END OF P1442 SUBPROGRAM   CSP21430

    NO ERRORS IN ABOVE ASSEMBLY.


// DUP                                                                  CSP21440
*STORE      WS  UA  P1442                                               CSP21450
33AF 0004
```

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

```
// ASM                                                                  CSP21460
** READ AND PUNCH SUBROUTINES FOR 1130 CSP                        (IO) CSP21470
* NAME READ                                                       (ID) CSP21480
* LIST                                                                 CSP21490
0053    19141100          ENT     READ       SUBROUTINE ENTRY POINT    CSP21500
                        * CALL READ (JCARD, J, JLAST, NERR1)           CSP21510
                        * READ COLUMNS FROM BEGINNING OF CARD INTO JCARD(J)  CSP21520
                        * THRDUGH JCARD(JLAST). PUT ERROR PARAMETER IN CSP21530
                        * NERR1.                                       CSP21540
008C    179150C8          ENT     PUNCH      SUBROUTINE ENTRY POINT    CSP21550
                        * CALL PUNCH (JCARD, J, JLAST, NERR2)          CSP21560
                        * PUNCH JCARD(J) THRDUGH JCARD(JLAST) INTO THE CSP21570
                        * BEGINNING OF A CARD. PUT ERROR PARAMETER INTO CSP21580
                        * NERR2.                                       CSP21590
0000 0  0000      JCARD DC      *-*        JCARD J  ADDRESS          CSP21600
0001    0051      AREA  BSS     81         I/O AREA BUFFER           CSP21610
0052 0  0000      FLAG  DC      *-*        ERROR INDICATOR           CSP21620
0053 0  0000      READ  DC      *-*        FIRST ARGUMENT ADDRESS    CSP21630
0054 0  691B            STX   1 SAVE1&1    SAVE IR1                  CSP21640
0055 01 65800053        LDX   I1 READ      GET 1ST ARGUMENT ADDRESS  CSP21650
0057 0  4022            BSI     SETUP      GO TO SETUP               CSP21660
0058 20 03059131        LIBF    CARD1      CALL CARD READ ROUTINE    CSP21670
0059 0  1000            DC      /1000      READ                      CSP21680
005A 1  0001            DC      AREA       AREA PARAMETER            CSP21690
005B 1  0073            DC      ERROR      ERROR PARAMETER           CSP21700
005C 20 225C5144  CONVT LIBF    SPEED      CALL CONVERSION ROUTINE   CSP21710
005D 0  0010            DC      /0010      CARD CODE TO EBCDIC       CSP21720
005E 1  0002            DC      AREA&1     FROM AREA                 CSP21730
005F 0  0000      JLAS1 DC      *-*        TO JCARD JLAST            CSP21740
0060 0  0000      CNT1  DC      *-*        CHARACTER COUNT           CSP21750
0061 0  C0F0            LD      FLAG       ERROR INDICATOR           CSP21760
0062 01 4C180067        BSC  L  FINAL,&-   ALL DONE IF ZERO          CSP21770
0064 0  1810            SRA     16         CLEAR ACC                 CSP21780
0065 0  D0EC            STO     FLAG       CLEAR THE INDICATDR       CSP21790
0066 0  70F5            MDX     CONVT      CONVERT AGAIN             CSP21800
0067 20 22989547  FINAL LIBF     SWING      REVERSE THE ARRAY         CSP21810
0068 1  0000            DC      JCARD      FROM JCARD J              CSP21820
0069 1  005F            DC      JLAS1      TO JCARD JLAST            CSP21830
006A 20 03059131  TEST  LIBF    CARD1      CALL BUSY TEST ROUTINE    CSP21840
006B 0  0000            DC      /0000      BUSY TEST PARAMETER       CSP21850
006C 0  70FD            MDX     TEST       REPEAT IF BUSY            CSP21860
006D 0  7104            MDX   1 4          INCREMENT 4 ARGUMENTS     CSP21870
006E 0  6903            STX   1 00NE&1     STORE IR1                 CSP21880
006F 00 65000000  SAVE1 LDX   L1 *-*       RESTORE IR1               CSP21890
0071 00 4C000000  DONE  BSC   L  *-*       RETURN TO CALLING PROGRAM CSP21900
0073 0  0000      ERROR DC      *-*        START OF ERROR ROUTINE    CSP21910
0074 00 D4000000  ERR   STD   L  *-*       STORE ACC IN ERROR WORD   CSP21920
0076 01 74010052        MDX   L  FLAG,1    SET THE FLAG INDICATOR    CSP21930
0078 01 4C800073        BSC   I  ERROR     RETURN TO INTERRUPT PROGRM CSP21940
007A 0  0000      SETUP DC      *-*        START OF SETUP ROUTINE    CSP21950
007B 20 01647880        LIBF    ARGS       CALL ARGS SUBPROGRAM      CSP21960
007C 1  0000            DC      JCARD      GET JCARD J  ADDRESS      CSP21970
007D 1  005F            DC      JLAS1      GET JCARD JLAST  ADDRESS  CSP21980
007E 1  0001            OC      AREA       GET CHARACTER COUNT       CSP21990
007F 0  0050            DC      80         MAX CHARACTER COUNT       CSP22000
0080 0  C0DE            LD      JLAS1      DISTRIBUTE JCARD JLAST    CSP22D10
0081 0  D014            STO     JLAS2      INTO JLAS2                CSP22020
```



```
0082 01 C4000001        LO    L  AREA      DISTRIBUTE COUNT          CSP22030
0084 0  00DB            STO     CNT1       INTO CNT1                 CSP22040
0085 0  0012            STO     CNT2       AND CNT2                  CSP22050
0086 0  C103            LO    1 3          GET ERROR WORD ADDRESS    CSP22060
0087 0  D0ED            STO     ERR&1      STORE INSIDE ERROR ROUTINE CSP22070
0088 0  1810            SRA     16         CLEAR ACC                 CSP22080
0089 0  D0C8            STO     FLAG       CLEAR ERROR INDICATOR     CSP22090
008A 01 4C80007A        BSC   I  SETUP     RETURN TO CALLING PROG    CSP22100
008C 0  0000      PUNCH DC      *-*        PUNCH ROUTINE STARTS HERE CSP22110
008D 0  69E2            STX   1 SAVE1&1    SAVE IR1                  CSP22120
008E 01 6580008C        LDX   I1 PUNCH     LOAD 1ST ARGUMENT ADDRESS CSP22130
0090 0  40E9            BSI     SETUP      GO TO SETUP ROUTINE       CSP22140
0091 20 22989547        LIBF    SWING      CALL REVERSE ARRAY        CSP22150
0092 1  0000            DC      JCARD      FROM JCARD J              CSP22160
0093 1  005F            DC      JLAS1      TO JCARD JLAST            CSP22170
0094 20 225C5144        LIBF    SPEED      CALL CONVERSION ROUTINE   CSP22180
0095 0  0011            OC      /0011      FROM EBCDIC TO CARD CODE  CSP22190
0096 0  0000      JLAS2 DC      *-*        FROM JCARD JLAST          CSP22200
0097 1  0002            DC      AREA&1     TO THE I/O AREA BUFFER    CSP22210
0098 0  0000      CNT2  DC      *-*        CHARACTER COUNT           CSP22220
0099 20 03059131        LIBF    CARD1      CALL PUNCH ROUTINE        CSP22230
009A 0  2000            OC      /2000      PUNCH                     CSP22240
009B 1  0001            OC      AREA       I/O AREA BUFFER           CSP22250
009C 1  0073            DC      ERROR      ERROR PARAMETER           CSP22260
009D 0  70C9            MDX     FINAL      ALL THROUGH, GO TO FINAL  CSP22270
009E                    END                END OF READ SUBPROGRAM    CSP22280

    NO ERRORS IN ABOVE ASSEMBLY.


// DUP                                                                 CSP22290
*STORE      WS  UA  READ                                               CSP22300
3383 0006
```

```
// ASM                                                                  CSP22310
** READ SUBRDUTINE FOR 1130 CSP, 2501                                  (IO) CSP22320
* NAME R2501                                                           (ID) CSP22330
* LIST                                                                 CSP22340
0053    19CB5C31          ENT     R2501       SUBROUTINE ENTRY PDINT     CSP22350
                  * CALL R2501(JCARD, J, JLAST, NERR1)                 CSP22360
                  * READ COLUMNS FROM BEGINNING OF CARD INTO JCARD(J)  CSP22370
                  * THROUGH JCARD(JLAST), PUT ERROR PARAMETER IN       CSP22380
                  * NERR1.                                            CSP22390
0000 0  0000      JCARO DC        *-*         JCARO J  ADORESS         CSP22400
0001    0051      AREA  BSS       81          I/O AREA BUFFER          CSP22410
0052 0  0000      FLAG  DC        *-*         ERRDR INDICATOR          CSP22420
0053 0  0000      R2501 DC        *-*         FIRST ARGUMENT ADDRESS   CSP22430
0054 0  692C              STX   1 SAVE1&1     SAVE IR1                 CSP22440
0055 01 65800053          LDX   I1 R2501      GET 1ST ARGUMENT ADDRESS CSP22450
0057 20 01647880          LIBF    ARGS        CALL ARGS SUBPROGRAM     CSP22460
0058 1  0000              DC      JCARD       GET JCARD J  AOORESS     CSP22470
0059 1  0072              OC      JLAS1       GET JCARO JLAST  ADDRESS CSP22480
005A 1  0001              DC      AREA        GET CHARACTER COUNT      CSP22490
005B 0  0050              DC      80          MAX CHARACTER COUNT      CSP22500
005C 0  C0A4              LD      AREA        OISTRIBUTE COUNT         CSP22510
005D 0  0015              STO     CNT1        INTO CNT1                CSP2252D
005E 0  C103              LD    1 3           GET ERROR WORD ADORESS   CSP22530
005F 0  D026              STD     ERR&1       STDRE INSIDE ERROR ROUTINE CSP2254D
0060 0  1810              SRA     16          CLEAR ACC                CSP22550
0061 0  D0F0              STO     FLAG        CLEAR ERROR INDICATDR    CSP22560
0062 0  7104              MDX   1 4           INCREMENT 4 ARGUMENTS    CSP22570
0063 0  691F              STX   1 OONE&1      STORE IR1                CSP22580
0064 0  C026              LO      ONE  SET AREA TD ALL ONES            CSP22590
0065 00 65000050          LDX   L1 80   LDAD IR1 WITH AREA SIZE        CSP22600
0067 01 D5000001    MD    STO   L1 AREA STDRE A ONE IN AREA            CSP22610
0069 0  71FF              MDX   1 -1   GO TD NEXT WORD DF AREA         CSP22620
006A 0  70FC              MDX     MO    GD BACK UNTIL FINISHED       CSP22630
006B 20 19141131          LIBF    READ1       CALL CARD READ ROUTINE   CSP22640
006C 0  1000              OC      /1000       READ                     CSP22650
006D 1  0001              OC      AREA        AREA PARAMETER           CSP22660
006E 1  0084              OC      ERRDR       ERROR PARAMETER          CSP22670
006F 20 225C5144    CONVT LIBF    SPEED       CALL CONVERSION ROUTINE  CSP22680
0070 0  0010              OC      /0010       CARO CDOE TO EBCDIC      CSP22690
0071 1  0002              OC      AREA&1      FRDM AREA                CSP22700
0072 0  0000      JLAS1 DC        *-*         TO JCARD JLAST           CSP22710
0073 0  0000      CNT1  DC        *-*         CHARACTER CDUNT          CSP22720
0074 0  C000              LD      FLAG        ERRDR INOICATDR          CSP22730
0075 01 4C18007A          BSC   L FINAL,&-    ALL OONE IF ZERO         CSP22740
0077 0  1810              SRA     16          CLEAR ACC                CSP22750
0078 0  00D9              STO     FLAG        CLEAR THE INDICATOR      CSP22760
0079 0  70F5              MDX     CDNVT       CONVERT AGAIN            CSP22770
007A 20 22989547    FINAL LIBF    SWING       REVERSE THE ARRAY        CSP22780
007B 1  0000              OC      JCARD       FROM JCARO J             CSP22790
007C 1  0072              DC      JLAS1       TO JCARD JLAST           CSP228D0
007O 20 19141131    TEST  LIBF    READ1       CALL BUSY TEST ROUTINE   CSP22810
007E 0  0000              OC      /0000       BUSY TEST PARAMETER      CSP22820
007F 0  70FO              MDX     TEST        REPEAT IF BUSY           CSP22830
0080 00 65000000    SAVE1 LDX   L1 *-*        RESTORE IR1              CSP22840
0082 00 4C000000    DDNE  BSC   L *-*         RETURN TD CALLING PROGRAM CSP22850
0084 0  0000        ERROR DC      *-*         START OF ERROR ROUTINE   CSP22860
0085 00 04000000    ERR   STO   L *-*         STORE ACC IN ERROR WORO  CSP22870
```



```
0087 01 74010052          MDX   L FLAG,1      SET THE FLAG INDICATDR   CSP22880
0089 01 4C800084          BSC   I ERROR       RETURN TO INTERRUPT PROGRM CSP22890
008B 0  0001        ONE   DC      1    CONSTANT OF ONE                 CSP22900
008C                      ENO               END OF R2501 SUBPROGRAM  CSP22910
```

NO ERRORS IN ABDVE ASSEMBLY.

```
// DUP                                                                  CSP22920
*STORE      WS  UA  R2501                                              CSP22930
3389 DD05
```

```
// ASM                                                                  CSP22940
** STACKER SELECT SUBROUTINE FDR 1130 COMMERCIAL SUBRDUTINE PACKAGE(ID) CSP22950
* NAME STACK                                                           (ID) CSP22960
* LIST                                                                 CSP22970
0002    228C1DD2          ENT     STACK STACK SUBROUTINE POINT         CSP22980
                  *                 CALL STACK                         CSP22990
                  *                 SELECTS THE NEXT CARO THROUGH      CSP23D00
                  *                 THE PUNCH STATION TD THE           CSP23D1D
                  *                 ALTERNATE STACKER ON THE 1442-5,   CSP23D2D
                  *                 6,OR 7.                            CSP23030
0000 0  0000      IDCC  DC      0     I/O COMMAND - FIRST WORD         CSP23040
0001 0  1480            DC      /1480 I/O COMMAND - SECONO WDRD        CSP23050
0002 0  0000      STACK DC      *-*   RETURN ADORESS COMES IN HERE     CSP23060
0003 0  08FC              XID     IOCC          SELECT STACKER         CSP23070
0004 01 4C800002          BSC   I STACK         RETURN TO CALLING PROG CSP23080
0006                      END                                          CSP23090
```

NO ERRDRS IN ABDVE ASSEMBLY.

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

```
// OUP                                                                  CSP23100
*STORE      WS  UA  STACK                                               CSP23110
338E 0002
```

```
// ASM                                                                  CSP23120
** TYPE ANO KEYBD SUBROUTINES FOR 1130 CSP                        (ID) CSP23130
* NAME TYPER                                                      (ID) CSP23140
* LIST                                                                 CSP23150
003F    23A17159          ENT     TYPER      SUBROUTINE ENTRY POINT    CSP23160
                        * CALL TYPE (JCARD, J, JLAST)                  CSP23170
                        * TYPE JCARD(J) THROUGH JCARO(JLAST)           CSP23180
0069    12168084          ENT     KEYBD      SUBROUTINE ENTRY POINT    CSP23190
                        * CALL KEYBO (JCARD, J, JLAST)                 CSP23200
                        * ENTER AT KEYBOARO JCARD(J) THROUGH JCARO(JLAST) CSP23210
0000 0  0001        ONE   OC      1          CONSTANT OF 1             CSP23220
0001 0  0000        JCARD OC      *-*        JCARD J  ADORESS          CSP23230
0002    0030        AREA  BSS     61         I/O AREA BUFFER           CSP23240
003F 0  0000        TYPER DC      *-*        FIRST ARGUMENT ADDR HERE  CSP23250
0040 0  691A              STX   1 SAVE1&1    SAVE IR1                  CSP23260
0041 0  6178              LDX   1 120        PUT 120 IN IR1            CSP23270
0042 0  6923              STX   1 MAXCH      STORE IT AS MAX CHARS     CSP23280
0043 01 6580003F          LDX   I1 TYPER     PUT FIRST ADDR IN IR1     CSP23290
0045 0  4018              BSI     SETUP      GO TO SETUP               CSP23300
0046 0  C0B8              LD      AREA       GET CHARACTER COUNT       CSP23310
0047 0  80B8              A       ONE        HALF AOJUST IT ANO        CSP23320
0048 0  1801              SRA     1          OIVIDE IT BY TWO          CSP23330
0049 0  D0B8              STO     AREA       AND REPLACE IT            CSP23340
004A 0  1001              SLA     1          OOUBLE IT                 CSP23350
004B 0  D008              STO     CNT1       AND PUT IT IN CNT1        CSP23360
004C 20 195C10D2          LIBF    RPACK      CALL REVERSE PACK ROUTINE CSP23370
004D 1  0001              DC      JCARO      FRDM JCARD J              CSP23380
004E 1  0083              DC      JLAST      TO JCARO JLAST            CSP23390
004F 1  0003              DC      AREA&1     PACK INTO I/O AREA        CSP23400
0050 2D 05097663          LIBF    EBPRT      CALL CDNVERSION ROUTINE   CSP23410
0051 0  0000              DC      /0000      FROM EBCOIC               CSP23420
0052 1  0003              DC      AREA&1     TO PRINTER COOE,          CSP23430
0053 1  0003              DC      AREA&1     ALL IN THE I/O AREA       CSP23440
0054 0  0000        CNT1  DC      *-*        HALF ADJSTD CHARACTER CNT CSP23450
0055 20 23A17170          LIBF    TYPEO      CALL TYPE ROUTINE         CSP23460
0056 0  2000              DC      /2000      TYPE PARAMETER            CSP23470
0057 1  0002              OC      AREA       I/O AREA BUFFER           CSP23480
0058 0  71D3        FINAL MDX   1 3          INCREMENT OVER 3 ARGUMENTS CSP23490
D059 0  6903              STX   I OONE&1     STORE IR1                 CSP23500
005A 00 65000000    SAVE1 LDX   L1 *-*       RESTORE IR1               CSP23510
005C 00 4C000000    OONE  BSC   L  *-*       RETURN TO CALLING PROGRAM CSP23520
005E 0  0000        SETUP OC      *-*        START OF SETUP ROUTINE    CSP23530
005F 20 23A17170    TEST  LIBF    TYPEO      CALL BUSY TEST RDUTINE    CSP23540
0060 0  0000              DC      /0000      BUSY TEST PARAMETER       CSP23550
0061 0  70F0              MDX     TEST       REPEAT TEST IF BUSY       CSP23560
0062 20 01647880          LIBF    ARGS       CALL ARGS ROUTINE         CSP23570
0063 1  0001              OC      JCARD      1ST ARGUMENT TO JCARD J   CSP23580
0064 1  0083              DC      JLAST      TO JCARO JLAST            CSP23590
0065 1  0002              DC      AREA       TO CHARACTER COUNT        CSP23600
0066 0  0000        MAXCH OC      *-*        MAXIMUM NUMBER OF CHARS   CSP23610
0067 01 4C80005E          BSC   I  SETUP      END OF SETUP, RETURN      CSP23620
0069 0  0000        KEYBD OC      *-*        START OF KEYBOARO ROUTINE CSP23630
006A 0  69F0              STX   1 SAVE1&1    SAVE IR1                  CSP23640
006B 0  613C              LOX   1 60         PUT BUFFER LENGTH IN IR1  CSP23650
006C 0  69F9              STX   1 MAXCH      60 IS MAX NO OF CHARS     CSP23660
006D 01 65800069          LDX   I1 KEYBD     1ST ARGUMENT ADDR IN IR1  CSP23670
006F 0  40EE              BSI     SETUP      GO TO SETUP               CSP23680
```



```
0070 0  613C              LDX   I 60         PUT BUFFER LENGTH IN IR1  CSP23690
0071 0  1810              SRA     16         CLEAR THE ACC             CSP23700
0072 01 05000002    CLEAR STO   L1 AREA      CLEAR THE I/O BUFFER      CSP23710
0074 0  71FF              MDX   1 -1         OECREMENT IR1             CSP23720
0075 0  70FC              MDX     CLEAR      ANO CONTINUE CLEARING     CSP23730
0076 01 65800069          LDX   I1 KEYBO     1ST ARGUMENT ADDR IN IR1  CSP23740
0078 0  C089              LO      AREA       PUT CHARACTER COUNT       CSP23750
0079 0  D00A              STO     CNT2       IN CNT2                   CSP23760
007A 20 23A17170          LIBF    TYPEO      CALL KEYBOARO ROUTINE     CSP23770
007B 0  1000              OC      /1000      KEYBOARD PARAMETER        CSP23780
007C 1  0002              OC      AREA       I/O AREA BUFFER           CSP23790
007D 20 23A17170    TEST1 LIBF    TYPEO      CALL BUSY TEST ROUTINE    CSP23800
007E 0  0000              OC      /0000      BUSY TEST PARAMETER       CSP23810
007F 0  70FD              MDX     TEST1      REPEAT TEST IF BUSY       CSP23820
0080 20 225C5144          LIBF    SPEEO      CALL CONVERSION ROUTINE   CSP23830
0081 0  0010              OC      /0010      CARO COOE TO EBCDIC       CSP23840
0082 1  0003              OC      AREA&1     FROM THE I/O AREA BUFFER  CSP23850
0083 0  0000        JLAST OC      *-*        TO JCARO JLAST            CSP23860
0084 0  0000        CNT2  OC      *-*        CHARACTER COUNT           CSP23870
0085 20 22989547          LIBF    SWING      CALL REVERSE ARRAY        CSP23880
0086 1  0001              OC      JCARO      REVERSE FROM JCARO J      CSP23890
0087 1  0083              OC      JLAST      TO JCARD JLAST            CSP23900
0088 0  70CF              MDX     FINAL      ALL THROUGH, GO TO FINAL  CSP23910
008A                      END                END OF TYPE SUBPROGRAM    CSP23920

   NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                              CSP23930
*STORE      WS  UA  TYPER                                           CSP23940
33C0 0006
```

```
// ASM                                                              CSP23950
** PACK/UNPAC SUBROUTINES FOR 1130 COMMERCIAL SUBROUTINE PACKAGE   (ID) CSP23960
* LIST                                                             CSP23970
* NAME UNPAC                                                       (ID) CSP23980
0000     24557043           ENT      UNPAC UNPACK SUBROUTINE ENTRY POINT  CSP23990
                    *                CALL UNPAC(JCARD,J,JLAST,KCARD,K)    CSP24000
                    *                THE WORDS JCARD J  THROUGH           CSP24010
                    *                JCARD JLAST  IN A2 FORMAT ARE        CSP24020
                    *                UNPACKED INTO XCARD K  IN A1 FORMAT. CSP24030
0006     17043480           ENT      PACK PACK SUBROUTINE ENTRY POINT     CSP24040
                    *                CALL PACK(JCARD,J,JLAST,KCARD,K)     CSP24050
                    *                THE WORDS JCARD J  THROUGH           CSP24060
                    *                JCARD JLAST  IN A1 FORMAT ARE PACKED CSP24070
                    *                INTO KCARD K  IN A2 FORMAT.          CSP24080
0000 0   0000       UNPAC DC         *-*  ARGUMENT ADDRESS COMES IN HERE  CSP24090
0001 0   C003             LD         SW2 LOAD NOP INSTRUCTION             CSP24100
0002 0   D01E             STO        SWTCH STORE NOP AT SWITCH            CSP24110
0003 0   7007             MDX        START COMPUTING                      CSP24120
0004 0   7009       SW1   MDX  X     ELSE-SWTCH-1 BRANCH TO ELSE          CSP24130
0005 0   7000       SW2   MDX  X     0 NOP INSTRUCTION                    CSP24140
0006 0   0000       PACX  DC         *-*  ARGUMENT ADDRESS COMES IN HERE  CSP24150
0007 0   C0FE             LD         PACK PICK UP ARGUMENT ADDRESS        CSP24160
0008 0   D0F7             STO        UNPAC AND STORE IT IN UNPAC          CSP24170
0009 0   C0FA             LD         SW1 LOAD BRANCH TO ELSE              CSP24180
000A 0   D016             STO        SWTCH STORE BRANCH AT SWITCH         CSP24190
000B 0   6930       START STX    1   SAVE1&1 SAVE IR1                     CSP24200
000C 01  65800000           LDX   I1  UNPAC PUT ARGUMENT ADDRESS IN IR1    CSP24210
000E 0   C100               LD     1  0 GET JCARD ADDRESS                 CSP24220
000F 0   8001               A         ONE+1 ADD CONSTANT OF 1              CSP24230
0010 00  95800001    ONE    S     I1  1 SUBTRACT J VALUE                   CSP24240
0012 0   D00D               STO       JCARD+1 CREATE JCARD(J) ADDRESS      CSP24250
0013 0   C103               LD     1  3 GET KCARD ADDRESS                 CSP24260
0014 0   80FC               A         ONE+1 ADD CONSTANT OF 1              CSP24270
0015 00  95800004           S     I1  4 SUBTRACT K VALUE                   CSP24280
0017 0   D006               STO       XCARD+1 CREATE KCARD(K) ADDRESS      CSP24290
0018 0   C100               LD     1  0 GET JCARD ADDRESS                 CSP24300
0019 0   80F7               A         ONE+1 ADD CONSTANT OF 1              CSP24310
001A 00  95800002           S     I1  2 SUBTRACT JLAST VALUE               CSP24320
001C 0   D0E9               STO       PACK CREATE JCARD JLAST  ADDRESS     CSP24330
001D 00  65000000    KCARD  LDX   L1  *-* PUT XCARD ADDRESS IN IR1         CSP24340
001F 00  C4000000    JCARD  LD    L   *-* PICK UP JCARD(J)                 CSP24350
0021 0   7000       SWTCH  MDX   X     0 SWITCH BETWEEN PACK AND UNPACK     CSP24360
0022 0   1888              SRT        8 SHIFT LOW ORDER BITS TO EXT        CSP24370
0023 0   1008              SLA        8 REPOSITION HIGH ORDER BITS         CSP24380
0024 0   E81A              OR         8MASK PUT BLANK IN LOW ORDER BITS    CSP24390
0025 0   D100              STO    1   0 PUT IN KCARD K                     CSP24400
0026 0   71FF              MDX    1   -1 DECREMENT KCARD ADDRESS           CSP24410
0027 0   1088              SLT        8    MOVE THE EXTEN INTO THE ACCUM   CSP24420
0028 0   1008              SLA        8    IN TWO STEPS                    CSP24430
0029 0   E815              OR         BMASK PUT BLANK IN LOW ORDER BITS    CSP24440
002A 0   7006              MDX        FINIS BRANCH AROUND PACK ROUTINE     CSP24450
0028 0   1898       ELSE   SRT        24 SHIFT HIGH ORDER BITS INTO EXT    CSP24460
002C 01  74FF0020          MDX   L    JCARD+1,-1 DECREMENT JCARD ADDRESS   CSP24470
002E 01  C4800020          LD    I    JCARD+1 PICK UP JCARD(J+1)           CSP24480
0030 0   18C8              RTE        8 SHIFT IN BITS FROM EXT             CSP24490
0031 0   D100       FINIS  STO    1   0 PUT IN KCARD K                     CSP24500
0032 01  74FF0020          MDX   L    JCARD+1,-1 DECREMENT JCARD ADDRESS   CSP24510
```



```
0034 0   71FF              MDX    1   -1 DECREMENT XCARD ADDRESS           CSP24520
0035 0   C0EA              LD         JCARD+1 GET JCARD(J) ADDRESS         CSP24530
0036 0   90CF              S          PACK SUBTRACT JCARD JLAST  ADDRESS    CSP24540
0037 01  4C10001F          BSC   L    JCARD,- CONTINUE IF DIFFERENCE & OR   CSP24550
0039 01  74050000          MDX   L    UNPAC,5 CREATE RETURN ADDRESS        CSP24560
0038 00  65000000    SAVE1 LDX   L1   *-* RESTORE IR1                      CSP24570
003D 01  4C800000          BSC   I    UNPAC RETURN TO CALLING PROGRAM      CSP24580
003F 0   0040        8MASK DC         /40 MASK 0000000001000000            CSP24590
0040                       END                                            CSP24600
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                              CSP24610
*STORE      WS  UA  UNPAC                                           CSP24620
33C6 0005
```

ADD
A1A3
A1DEC
A3A1
CARRY
DECA1
DIV
DPACK
DUNPK
EDIT
FILL
GET
ICOMP
IOND
KEYBD
MOVE
MPY
NCOMP
NSIGN
NZONE
PACK
PRINT
PUNCH
PUT
P1403
P1442
READ
R2501
SKIP
STACK
SUB
S1403
TYPER
UNPAC
WHOLE

```
// ASM                                                                  CSP24630
** WHOLE NUMBER SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE  (ID) CSP24640
* NAME WHOLE                                                          (ID) CSP24650
* LIST                                                                 CSP24660
0006     262164C5           ENT     WHOLE      SUBROUTINE ENTRY POINT    CSP24670
                    * X=WHOLE(Y), WITH Y IN FAC TO START                 CSP24680
                    * X IN FAC BECOMES THE INTEGRAL PART OF Y.           CSP24690
0000 0   0000       DBL1    DC      0          DBL CONSTANT OF 1         CSP24700
0001 0   0001               DC      1          REST OF DBL1 CONSTANT     CSP24710
001F             MANT    EQU     31         MANTISSA LENGTH           CSP24720
0002 0   009F       C159    DC      128+MANT   EXPONENT OF FULL INTEGER  CSP24730
0003 0   001F       C31     DC      MANT       MANTISSA LENGTH           CSP24740
0004 0   189F       SRT     SRT     MANT       SRT MANTISSA LENGTH       CSP24750
0005 0   0800       H0800   DC      /0800      DIFF BETWEEN SRT AND SLT  CSP24760
0006 0   0000       WHOLE   DC      *-*        ARGUMENT ADDRESS HERE     CSP24770
0007 0   C0FA               LD      C159       EXP OF FULL INTEGER       CSP24780
0008 0   9370               S     3 125        SUBTRACT EXP OF Y         CSP24790
0009 01  4C28001A           BSC   L ONE,+Z     BRANCH IF ALL INTEGER     CSP24800
000B 0   90F7               S       C31        SUBTRACT MANTISSA LENGTH  CSP24810
000C 01  4C10001E           BSC   L FRACT,-    BRANCH IF ALL FRACTIONAL  CSP24820
000E 0   80F5               A       SRT        CREATE RIGHT SHIFT        CSP24830
000F 0   0005               STO     RIGHT      STORE  RIGHT SHIFT        CSP24840
0010 0   90F4               S       H0800      CREATE  LEFT SHIFT        CSP24850
0011 0   D006               STO     LEFT       STORE   LEFT SHIFT        CSP24860
0012 0   CB7E               LDD   3 126        PICK UP MANTISSA          CSP24870
0013 0   4828               BSC     +Z         CHECK FOR NEGATIVE MANTISA CSP24880
0014 0   98EB               SD      DBL1       SUBTRACT 1 IF NEGATIVE    CSP24890
0015 0   1880       RIGHT   SRT     *-*        RIGHT SHIFT               CSP24900
0016 0   4828               BSC     +Z         CHECK FOR NEGATIVE MANTISA CSP24910
0017 0   88E8               AD      DBL1       ADD 1 IF NEGATIVE         CSP24920
0018 0   1080       LEFT    SLT     *-*        LEFT SHIFT                CSP24930
0019 0   0B7E       STORE   STD   3 126        STORE MANTISSA            CSP24940
001A 01  74010006   DONE    MDX   L WHOLE+1    CREATE RETURN ADDRESS     CSP24950
001C 01  4C800006           BSC   I WHOLE      RETURN TO CALLING PROGRAM CSP24960
001E 0   10E0       FRACT   SLC     32         ZERO ACC AND EXT          CSP24970
001F 0   D37D               STO   3 125        ZERO THE EXPONENT         CSP24980
0020 0   70F8               MDX     STORE      ZERO THE MANTISSA         CSP24990
0022                        END                ENO OF WHOLE SUBROUTINE   CSP25000

    NO ERRORS IN ABOVE ASSEMBLY.

// DUP                                                                  CSP25010

*STORE      WS  UA  WHOLE                                               CSP25020

33CB 0003
```

```
// ASM                                                                  CSP25030
** ARGS, RPACK AND SWING SUBROUTINES FOR 1130 CSP                (ID) CSP25040
* LIST                                                                 CSP25050
* NAME ARGS                                                     (ID)   CSP25060
                        LIBR            LIBF TYPE ROUTINES FOLLOW    CSP25070
                    * THESE SUBROUTINES CANNOT BE CALLED FROM FORTRAN CSP25080
0002    01647880            ENT     ARGS        SUBROUTINE ENTRY POINT  CSP25090
                    * ARGS GETS THE ARGUMENT FOR THE I/O ROUTINES      CSP25100
0030    195C1002            ENT     RPACK       SUBROUTINE ENTRY POINT  CSP25110
                    * RPACK REVERSES AND PACKS EBCDIC STRINGS          CSP25120
004F    22989547            ENT     SWING       SUBROUTINE ENTRY POINT  CSP25130
                    * SWING REVERSES AN EBCDIC STRING                  CSP25140
0000 0  0001        ONE     DC      1           CONSTANT OF ONE         CSP25150
0001 0  0000        JLAST   DC      *-*         JCARD(JLAST) ADDRESS    CSP25160
0002 0  6A2A        ARGS    STX   2 SAVE2&1     ARGS ROUTINE STARTS HERE CSP25170
0003 00 66800000            LDX   I2 0          GET 1ST ARGUMENT ADDR   CSP25180
0005 0  C100                LD    1 0           GET JCARD ADDR          CSP25190
0006 00 95800002            S     I1 2          SUBTRACT JLAST VALUE    CSP25200
0008 0  80F7                A       ONE         ADD ONE                 CSP25210
0009 00 06800001            STO   I2 1          STORE IN 2ND ARG        CSP25220
0008 0  C100                LD    1 0           GET JCARD ADDR          CSP25230
000C 00 95800001            S     I1 1          SUBTRACT J VALUE        CSP25240
000E 0  80F1                A       ONE         ADD ONE                 CSP25250
000F 00 D6800000            STO   I2 0          STORE IN 1ST ARG        CSP25260
0011 00 96800001            S     I2 1          SUBTRACT JLAST ADDR     CSP25270
0013 0  80EC                A       ONE         ADD ONE                 CSP25280
0014 01 4C080018            BSC   L  ERDR1,+    CHECK FOR NEG OR 0 CHARS CSP25290
0016 0  9203                S     2 3           OK. SUBTRACT MAX CHARS  CSP25300
0017 01 4C300021            BSC   L  ERROR,-    CHECK MORE THAN MAX CHARS CSP25310
0019 0  8203                A     2 3           ADD MAX CHARS BACK      CSP25320
001A 0  7000                MDX     OK          ADDRESSES OK            CSP25330
001B 00 C6800000    ERDR1   LD    I2 0          PICK UP JCARD(J)        CSP25340
001D 00 D6800001            STO   I2 1          AND STORE IN JCARD(JLAST) CSP25350
001F 0  C0E0                LD      ONE         SET UP CHAR COUNT OF 1  CSP25360
0020 0  7007                MDX     OK          GO TO STORE CHAR COUNT  CSP25370
0021 00 C6800000    ERROR   LD    I2 0          PICK UP JCARD(J)        CSP25380
0023 0  9203                S     2 3           AND CALCULATE JCARD(JLAST) CSP25390
0024 0  80DB                A       ONE         TO BE JCARD(J+MAX-1)    CSP25400
0025 00 D6800001            STO   I2 1          STORE ADDR IN JCARD(JLAST) CSP25410
0027 0  C203                LD    2 3           LOAD CHARACTER COUNT    CSP25420
0028 00 D6800002    OK      STO   I2 2          STORE CHARACTER COUNT   CSP25430
002A 0  7204                MDX   2 4           CREATE RETURN ADDR      CSP25440
002B 0  6A03        LAST    STX   2 DONE&1      STORE RETURN ADDRESS    CSP25450
002C 00 66000000    SAVE2   LDX   L2 *-*        RESTORE IR2             CSP25460
002E 00 4C000000    ODNE    BSC   L  *-*        RETURN TO CALLING PROGRAM CSP25470
0030 0  6AFC        RPACK   STX   2 SAVE2&1     RPACK ROUTINE STARTS HERE CSP25480
0031 00 66800000            LDX   I2 0          GET 1ST ARGUMENT ADDRESS CSP25490
0033 00 C6800000            LD    I2 0          GET JCARD ADDR          CSP25500
0035 0  D006                STO     JCARD&1     INITIALIZE JCARD ADDRESS CSP25510
0036 00 C6800001            LD    I2 1          GET SECOND ARGUMENT ADDR CSP25520
0038 0  D0C8                STO     JLAST       INITIALIZE JCARD JLAST  CSP25530
0039 0  C202                LD    2 2           GET AREA ADDRESS        CSP25540
003A 0  0009                STO     KCARD&1     INITIALIZE PACK TO ADDRESS CSP25550
003B 00 C4000000    JCARD   LD    L  *-*        LOAD FIRST CHARACTER    CSP25560
003D 0  1898                SRT     24          SHIFT INTO EXT          CSP25570
003E 01 74FFD03C            MDX   L  JCARD&1,-1 DECREMENT ADDRESS       CSP25580
0040 01 C480003C            LD    I  JCARD&1    GET SECOND CHARACTER    CSP25590
```



```
0042 0  18C8                RTE     8           SHIFT RIGHT, RETRIEVE EXT CSP25600
0043 00 D4000000    KCARO   STO   L  *-*        STORE IN AREA           CSP25610
0045 01 74FF003C            MDX   L  JCARD&1,-1 DECREMENT ADDRESS       CSP25620
0047 01 74010044            MDX   L  KCARD&1,&1 INCREMENT AREA ADDRESS  CSP25630
0049 0  C0F2                LD      JCARD&1     GET ENDING ADDRESS      CSP25640
004A 0  90B6                S       JLAST       SUBTRACT JCARD JLAST  ADDR CSP25650
004B 01 4C10003B            BSC   L  JCARD,-    REPEAT IF NOT MINUS     CSP25660
004D 0  7203                MDX   2 3           INCREMENT OVER 3 ARGS   CSP25670
004E 0  700C                MDX     LAST        ALL THROUGH, GO TO LAST CSP25680
004F 0  6A00        SWING   STX   2 SAVE2&1     SWING ARRAY END FOR END CSP25690
0050 00 66800000            LDX   I2 0          GET 1ST ARGUMENT ADDRESS CSP25700
0052 00 C6800000            LD    I2 0          GET FIRST ARGUMENT      CSP25710
0054 0  D007                STO     BACK&1      STORE AT BACK ADDRESS   CSP25720
0055 00 C6800001            LD    I2 1          GET 2ND ARGUMENT        CSP25730
0057 0  D001                STO     FRONT&1     STORE AT FRONT ADDRESS  CSP25740
0058 00 C4000000    FRONT   LD    L  *-*        GET WORD FROM FRONT     CSP25750
005A 0  1890                SRT     16          PUT IT IN THE EXT       CSP25760
005B 00 C4000000    BACK    LD    L  *-*        GET A WORD FROM THE BACK CSP25770
005D 0  E810                OR      HEX40       OR IN AN EBCDIC BLANK   CSP25780
005E 01 D4800059            STO   I  FRONT&1    PUT IT IN THE FRONT     CSP25790
0060 0  1090                SLT     16          RETRIEVE THE EXT        CSP25800
0061 0  E80C                OR      HEX40       OR IN AN EBCDIC BLANK   CSP25810
0062 01 0480005C            STO   I  BACK&1     PUT IT IN THE BACK      CSP25820
0064 01 74010059            MDX   L  FRONT&1,&1 INCREMENT THE FRONT ADDR CSP25830
0066 01 74FF005C            MDX   L  BACK&1,-1  DECREMENT THE BACK ADDR  CSP25840
0068 0  C0F0                LD      FRONT&1     GET THE FRONT ADDRESS   CSP25850
0069 0  90F2                S       BACK+1      SUBTRACT THE BACK ADDRESS CSP25860
006A 01 4C080058            BSC   L  FRONT,&    REPEAT IF MINUS         CSP25870
006C 0  7202                MDX   2 2           INCREMENT OVER 2 ARGS   CSP25880
006D 0  70BD                MDX     LAST        ALL THROUGH, GO TO LAST CSP25890
006E 0  0040        HEX40   DC      /0040       EBCDIC BLANK CODE       CSP25900
0070                        END                 END OF ARGS SUBPROGRAM  CSP25910

    NO ERRORS IN ABOVE ASSEMBLY.
```

```
// DUP                                                                  CSP25920
*STORE      WS  UA  ARGS                                                CSP25930
33CE 0008
```

APPENDIX

CORE ALLOCATION

To calculate the core requirements, sum the number of words for all routines used. If NZONE, CARRY, NSIGN, SERVICE, WHOLE, ADD, and/or FILL are not included in the first sum, and they are CALLed by a routine in the first sum, add their number of words to the first sum. Then calculate the Reference core requirements. Keep in mind that no matter how many times a Reference is used, it should be considered only once. Sum the core requirements of all References used. Add this sum to the first sum. The resulting total is the core requirement for the 1130 Commercial Subroutine Package. Notice that the FORTRAN subroutines a, b, and c will be used by most FORTRAN programs and so will be present whether the package is used or not.

CSP Routine Name	Number of Words	Calls These CSP Routines	Calls These Subroutine Library Routines
A1DEC	74	NZONE	-
A1A3/A3A1	152	-	-
ADD/SUB	170	CARRY, FILL	-
ARGS	112	-	-
CARRY	54	-	-
DECA1	76	NZONE	-
DIV	238	CARRY, FILL	-
DPACK/DUNPK	100	-	-
EDIT	204	NZONE, FILL	-
FILL	30	-	-
GET	96	NZONE	ref. a and b
ICOMP	122	-	-
IOND	6	-	-
MOVE	36	-	-
MPY	164	CARRY, FILL	-
NCOMP	42	-	-
NSIGN	42	-	-
NZONE	78	-	-
PACK/UNPAC	66	-	-
PRINT/SKIP	124	ARGS	ref. e
PUT	104	NZONE, WHOLE	ref. a, b, and c
P1403/S1403	134	ARGS	ref. j
P1442	130	ARGS	ref. i
READ/PUNCH	158	ARGS	ref. f and h
R2501	140	ARGS	ref. d and h
STACK	6	-	-
TYPER/KEYBD	138	ARGS	ref. g and h
WHOLE	34	-	-

References

a. (EADD, EMPY, ESTO, FLOAT, NORM) 342 words
b. (SNR) 8 words
c. (EABS, IFIX) 74 words
d. (READ1) 110 words
e. (PRNT1) 404 words
f. (CARD1) 264 words
g. (TYPE0, EBPRT) 638 words
h. (SPEED, ILS04) 360 words
i. (PNCH1) 218 words
j. (PRNT3, ZIPCO, EBPT3) 544 words

EBCDIC CHARACTERS AND DECIMAL EQUIVALENTS

A	-16064	S	-7616	blank	16448
B	-15808	T	-7360	. (period)	19264
C	-15552	U	-7104	< (less than)	19520
D	-15296	V	-6848	(	19776
E	-15040	W	-6592	+	20032
F	-14784	X	-6336	&	20544
G	-14528	Y	-6080	$	23360
H	-14272	Z	-5824	*	23616
I	-14016	0	-4032	)	23872
J	-11968	1	-3776	- (minus)	24640
K	-11712	2	-3520	/	24896
L	-11456	3	-3264	,	27456
M	-11200	4	-3008	%	27712
N	-10944	5	-2752	#	31552
O	-10688	6	-2496	@	31808
P	-10432	7	-2240	' (apostrophe)	32064
Q	-10176	8	-1984	=	32320
R	-9920	9	-1728		

Subprogram Name	Approximate* Execution Time in Microseconds**
GET	2250 + 2190 C
PUT	3450 + 3090 C
EDIT	630 + 90 S + 180 M
MOVE	300 + 45 C
FILL	300 + 30 C
WHOLE	1400
NCOMP	250 + 75 C
NZONE	350
ICOMP	500 + 95 C
NSIGN	240
ADD	2160 + 216 L
SUB	2160 + 216 L
MPY	2400 + 120 P
DIV	4000 + Q (445 + 667 DIV)
A1DEC	700 + 54 A
DECA1	180 + 117 A
A1A3	470 + 1084 A
A3A1	545 + 156 A
PACK	360 + 63 A
UNPAC	420 + 66 A
DPACK	392D
DUNPK	360D

C = Length of the field, in characters
S = Length of the source field
M = Length of the edit mask
P = Length of the multiplier field x length of the multiplicand field (significant digits only--don't count leading zeros)
A = Length of the A1 field
D = Length of the packed decimal (D4) field
L = Length of the longer of the two fields (significant digits only--don't count leading zeros)
Q = Number of significant digits in the quotient (result) field
DIV = Number of significant digits in the divisor (denominator) field

* All timings are approximate, and are based on test runs of "typical" cases, using fields of "average" size, magnitude, etc. Unusual cases may (or may not) differ significantly from the timings obtained from the given equations. This is particularly true of the decimal arithmetic routines (ADD, SUB, MPY, DIV).

** Based on 3.6-microsecond CPU cycle speed. Multiply by 0.6 to obtain timings on 2.2-microsecond CPU.

This page intentionally left blank.

Format of Commercial Subroutine Calls (and Parameters*)	Page Nos.**	Format of Data: Before		Format of Data: After	Comments on Parameters
*ONE WORD INTEGERS		—	---	—	Must use for every CSP program
*EXTENDED PRECISION		—	---	—	Must use if GET or PUT is present
*IOCS (DISK)		—	---	—	Only DISK can be specified for CSP I/O
CALL ADD(JCARD,J,JLAST,KCARD,K,KLAST,NER)	13	D1	---	D1	Initialize NER to 0; error if NER=KLAST
CALL A1A3(JCARD,J,JLAST,KCARD,K,ICHAR)	15	A1	---	A3	You must define ICHAR array, and it must contain 40 characters
CALL A1DEC(JCARD,J,JLAST,NER)	18	A1	---	D1	Initialize NER to 0; error if NER≠0
CALL A3A1(JCARD,J,JLAST,KCARD,K,ICHAR)	21	A3	---	A1	You must define ICHAR array, and it must contain 40 characters
CALL DECA1(JCARD,J,JLAST,NER)	26	D1	---	A1	Initialize NER to 0; error if NER≠0
CALL DIV(JCARD,J,JLAST,KCARD,K,KLAST,NER)	28	D1	---	D1	Initialize NER to 0; error if NER=KLAST
CALL DPACK(JCARD,J,JLAST,KCARD,K)	31	D1	---	D4	
CALL DUNPK(JCARD,J,JLAST,KCARD,K)	34	D4	---	D1	
CALL EDIT(JCARD,J,JLAST,KCARD,K,KLAST)	36	A1	---	A1	Control characters in mask are: b0.,CR-*S
CALL FILL(JCARD,J,JLAST,NCH)	41	Dec.	---	A1	See reverse side for decimal values for NCH
GET(JCARD,J,JLAST,SHIFT)	42	A1	---	Real***	SHIFT must be real, extended precision. (1.0=no shift)
ICOMP(JCARD,J,JLAST,KCARD,K,KLAST)	45	A1	---	-0+	Minus:JCARD<KCARD;Zero:JCARD=KCARD;Plus:JCARD>KCARD.
CALL IOND	47	None	---	None	Use before PAUSE or STOP (Monitor Version 1 Only)
CALL KEYBD(JCARD,J,JLAST)	48	A1	---	A1	Maximum of 60 Characters allowed
CALL MOVE(JCARD,J,JLAST,KCARD,K)	50	Any	---	Same	
CALL MPY(JCARD,J,JLAST,KCARD,K,KLAST,NER)	52	D1	---	D1	Initialize NER to 0; error if NER=KLAST
NCOMP(JCARD,J,JLAST,KCARD,K)	54	A1	---	-0+	Minus:JCARD<KCARD;Zero:JCARD=KCARD;Plus:JCARD>KCARD.
CALL NSIGN(JCARD,J,NEWS,NOLDS)	56	D1		Integer	See reverse side for values for NEWS and NOLDS
CALL NZONE(JCARD,J,NEWZ,NOLDZ)	58	A1	---	Integer	See reverse side for values for NEWZ and NOLDZ
CALL PACK(JCARD,J,JLAST,KCARD,K)	60	A1	---	A2	
CALL PRINT(JCARD,J,JLAST,NER)	62	A1	---	A1	Initialize NER to 0; if NER=3, reached chan.9; if NER=4, reached chan. 12
CALL PUNCH(JCARD,J,JLAST,NER)	64	A1	---	A1	Initialize NER to -1; if NER=0, last card, if NER=1, feed or punch check
CALL PUT(JCARD,J,JLAST,VAR,ADJST,N)	66	Real***	---	A1	VAR and ADJST must be real, extended precision
CALL P1403(JCARD,J,JLAST,NER)	68	A1	---	A1	Initialize NER to 0; if NER=3, reached chan. 9; if NER=4, reached chan. 12
CALL P1442(JCARD,J,JLAST,NER)	70	A1	---	A1	Initialize NER to -1; if NER=0, last card; if NER=1, feed or punch check
CALL READ(JCARD,J,JLAST,NER)	73	A1	---	A1	Initialize NER to -1; if NER=0, last card; if NER=1, feed or read check
CALL R2501(JCARD,J,JLAST,NER)	76	A1	---	A1	Initialize NER to -1; if NER=0, last card; if NER=1, feed or read check
CALL SKIP(N)	79	Dec.	---	None	See reverse side for functional values for N
CALL S1403(N)	84	Dec.	---	None	See reverse side for functional values for N
CALL STACK	81	None	---	None	
CALL SUB(JCARD,J,JLAST,KCARD,K,KLAST,NER)	82	D1	---	D1	Initialize NER to 0; error if NER=KLAST
CALL TYPER(JCARD,J,JLAST)	86	A1	---	A1	See reverse side for values for functional characters
CALL UNPAC(JCARD,J,JLAST,KCARD,K)	89	A2	---	A1	
WHOLE(EXPRESSION)	91	Real	---	Real	The expression must be "real" not "integer".

* All parameters required by each subroutine must be supplied.

** Page Number in 1130 Commercial Subroutine Package (1130-SE-25X), Version 3 Program Reference Manual (H20-0241-3)

*** Must use extended precision in calling program.

FILL and NCOMP

	EBCDIC Char.	Dec. Equiv.
Low	(12-0)	-16320
↑ Listed in Collating Sequence	A	-16064
	B	-15808
	C	-15552
	D	-15296
	E	-15040
	F	-14784
	G	-14528
	H	-14272
	I	-14016
	(11-0)	-12224
	J	-11968
	K	-11712
	L	-11456
	M	-11200
	N	-10944
	O	-10688
	P	-10432
	Q	-10176
	R	-9920
	S	-7616
	T	-7360
	U	-7104
	V	-6848
	W	-6592
	X	-6336
	Y	-6080
	Z	-5824
	0	-4032
	1	-3776
	2	-3520
	3	-3264
	4	-3008
	5	-2752
	6	-2496
	7	-2240
	8	-1984
	9	-1728
	blank	16448
	. (periad)	19264
	< (less than)	19520
	(	19776
	+	20032
	&	20544
	$	23360
	*	23616
	)	23872
	- (minus)	24640
	/	24896
	,	27456
	%	27712
High	#	31552
	@	31808
	' (apostraphe)	32064
	=	32320

NSIGN — used with D1 fields

If NOLDS IS:	Then sign was:
+1	positive
-1	negative

When NEWS is:	Sign is set to:
+1	pasitive
0	opposite af old sign
-1	negative
NOLDS	na change

NZONE — used with A1 fields

If NOLDZ is:	Then character was:
1	A-I
2	J-R
3	S-Z
4	0-9
more than 4	special

When NEWZ is:	Character is set to:
1	12 zane
2	11 zane
3	0 zone
4	no zone
more than 4	no change

SKIP and S1403 function	Value for N
Immediate skip to channel 1	12544
Immediate skip ta channel 2	12800
Immediate skip to channel 3	13056
Immediate skip to channel 4	13312
Immediate skip ta channel 5	13568
Immediate skip ta channel 6	13824
Immediate skip to channel 9	14592
Immediate skip ta channel 12	15360
Immediate space af 1 space	15616
Immediate space af 2 spaces	15872
Immediate space af 3 spaces	16128
Suppress space after printing	0

Normal spacing is ane space after printing.

TYPER function	Decimal constant in (JCARD) output area
Tabulate	1344
Shift to black	5184
Carrier return	5440
Backspace	5696
Line Feed	9536
Shift ta red	13632

The procedures set forth in IBM 1130 Card/Paper Tape Programming System Operator's Guide (C26-3629) and in IBM 1130 DISK Monitor System Reference Manual (C26-3750 or C26-3717) should be followed to execute the sample problems and all user-written programs.

Switch settings for the sample problems are as follows:

Input Device	Output Device	Switches		
		0	1	2
1442	console printer	down	down	down
1442	1132	up	down	down
1442	1403	up	up	down
2501	console printer	down	down	up
2501	1132	up	down	up
2501	1403	up	up	up

Make sure that the switches are set properly before the program begins.

Note: Sample Problem 2 cannot be executed if Version 1 of the Monitor is being used.

HALT LISTING

Conditions A and B (see list below) have the following meaning:

A Device not ready.

B Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listings in this manual. If the deck is the same, contact your local IBM representative. Save all output.

IAR	Accumulator (hex)	Device	Condition
41	1xx0	1442 Card Read Punch	A
41	1xx1	1442 Card Read Punch	B
41	2xx0	Console printer or keyboard	A
41	2xx1	Console printer or keyboard	B
41	4xx0	2501 Card Reader	A
41	4xx1	2501 Card Reader	B
41	6xx0	1132 Printer	A
41	6xx1	1132 Printer	B
41	9xx0	1403 Printer	A
41	9xx1	1403 Printer	B

BIBLIOGRAPHY

IBM 1130 Functional Characteristics (A26-5881)

Core Requirements for 1130 FORTRAN (C20-1641)

1130 FORTRAN Programming Techniques (C20-1642)

IBM 1130 Card/Paper Tape Programming System Operator's Guide (C26-3629)

IBM 1130 DISK Monitor System Reference Manual (C26-3750)

IBM 1130 Assembler Language (C26-5927)

IBM 1130 Subroutine Library (C26-5929)

IBM 1130/1800 Basic FORTRAN IV Language (C26-3715)

IBM 1130 DISK Monitor System, Version 2 (C26-3717)

READER'S COMMENT FORM

H20-0241-3

1130 Commercial Subroutine Package
(1130-SE-25X), Version 3
Program Reference Manual

Please comment on the usefulness and readability of this publication, suggest additions and delctions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold fold

fold fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold

fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold

fold

Printed in U.S.A. H20-0241-3

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Printed in U. S. A. H20-0241-3

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)